Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 33 – a.a 2020/2021

TESI DI DOTTORATO / PH.D. THESIS

Intelligent privacy safeguards for the digital society

ALFONSO GUARINO

SUPERVISOR: **PROF. DELFINA MALANDRINO**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

[illegible]

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata
Dipartimento di Informatica

[illegible]

Abstract

The growth of the Internet and the pervasiveness of Information and Communication Technology (ICT) have led to a radical change in our society, a deep economical, commercial and social impact on our lives. To date, most of our lives takes place online where algorithms shape and guide our behaviour and the governance of our societies.

One of the drawbacks of this change is an increased risk for Internet users about their personal information *privacy*. Indeed an enormous amount of data is being generated and disseminated by people at high pace, often without knowing who is recording what about them. Online browsing, banking, shopping, social network interactions, and any type of online economic, social, personal collaboration and communication could undermine the individuals' privacy due to a variety of factors that include not only the frightening increase of information leakage. Indeed, specific private information can be also inferred/extracted via computational heuristics applied on data (apparently unrelated to such information) users voluntarily disclose on the Internet.

In particular, such privacy leaks can be caused by both *(a)* applications or software users intentionally use unaware of the related risks, and *(b)* malicious (illegal or unfair) practices stealthy perpetrated by "adversaries". Therefore, securing private data, devices and user's privacy in the digital society has become an utmost concern for individuals, business organizations, national governments and researchers.

Given the complexity, inscrutability of the software users engage with and the number of potential attacks and unfair practices they can incur into, it becomes ever more clear the need for technology-driven safeguards supporting users in the detection and counteract of such threats. In this respect, *machine learning* (ML), with its pattern recognition capability, appears to be a precious ally capable of upholding not only users' privacy but also other rights threatened in the digital society.

In this dissertation, we *focus* on intelligent *privacy* safeguards for users in the digital society. Our *goal* is to explore, both on the theoretical and experimental levels, how ML approaches can be fit to support the protection of privacy and the related rights. The research draws also upon most recent development in the areas of computational law and techno-regulation, two research paradigms emerging, at planetary scale, on the boundaries between computer science and law.

The work is structured as follows. We first depict the theoretical and methodological framework of this work, through a systematic literature review framing the use of ML to protect users' privacy. In doing this, we trace back existing approaches and solutions for the privacy protections to two fundamental categories: *enforcement* (i.e., solutions which *impose constrains* and hamper breaches of norms) and *nudge* solutions (i.e., solutions which inform users and increase their *awareness* to promote privacy-oriented behaviors). We provide a comprehensive taxonomy of main areas, threats, ML methods, type of protection delivered analyzing 143 studies published from January 2017 to October 2020.

Then, we present a series of research activities exploring the applicability of ML-based approaches to the issues arising in the scenarios above described. The activities presented can be ideally split into two parts focusing on privacy protection and other related rights, respectively. In more detail, the first part encompasses two projects tackling the privacy protection in the strict sense.

a) ML for privacy enforcement

We deal with the long-standing issue of third-party tracking on the Web in which users' private data are unfairly stolen for marketing

and malicious activities, such as online stalking. We experiment the use of ML to distinguish between trackers and functional resources on the Web, finding that such techniques can be fit for a high classification rate of such threat. The resulting ML-based approach has been implemented into *GuardOne*, a tool to protect users against third-party tracking, which provides *enforcement* solutions to block trackers. *GuardOne* has been evaluated in real-world against similar commercial privacy enforcing solution. The main features of *GuardOne* are: *(i)* a hybrid mechanism based on ML and blacklisting, *(ii)* customization based on the user browsing habits, *(iii)* a very lightweight implementation which does not impact on the users' devices performance compared to commercial solutions, *(iv)* a high effectiveness in detecting and blocking third-party trackers better than the vast majority of commercial solutions.

b) ML for privacy awareness
We deal with the issue of unaware and/or uncontrolled dissemination of personal and private data, in text format, on the Internet. We experiment the use of ML and advanced language processing techniques to support both the classification of the text topic (among the most sensitive ones, e.g., politics and health) and the sensitiveness of the content according to such topic, finding that the performance of our proposal in a simulated environment is comparable with solutions available in literature. Furthermore, we experiment how the ML solutions designed can be fit to learn the user's personal attitudes towards privacy. We then embed such ML-approaches into *Knoxly*, a tool to protect users against the dissemination of personal and/or private information online, which relies on *nudge*-based solutions. Specifically, *Knoxly* aims to raise awareness and promote privacy-oriented behavior by means of alerts/warnings. The main features of *Knoxly* are: *(i)* a Keyword module to detect common sensitive words and personal identifiable information, *(ii)* a Topic module to distinguish the text's topic, *(iii)* a Sensitiveness module to "measure" the sensitiveness of the text content, *(iv)* a Customized module allowing the user to personalize the warnings displayed, *(v)* an intuitive User In-

terface powered by Visual Analytics techniques, *(vi)* a lightweight implementation which does not impact on the users' devices performance.

The second part encompasses two other projects that exploit methodological and technological solutions and approaches identified in the previous research stage and expand the research scope by applying such insights to other rights, linked to privacy and of great relief in the digital society, i.e., child and consumer protection.

a) ML for child protection
We deal with the challenge of providing online (privacy) protections for a specific category of users, that is children, which, according to the General Data Protection Regulation (GDPR) and UNICEF, need *ad-hoc* safeguards. Within this research project, named *AI4Children*, we experiment several ML-based approaches for users identification which can be seen as the baseline to uphold legal standards for online child protection. In fact, once identified the user, it is possible to trigger the specific safeguards. In more details, the conceived approach (based on data integration techniques) is capable of recognizing the age of a user based on the touch gestures he/her performs on a mobile device with a high accuracy. The main features of *AI4Children* are: *(i)* distinguishing between adults and underages based on commonly performed touch gestures, *(ii)* using a small set of features and touch gesture to perform a high accurate classification, *(iii)* robustness in the classification on different devices.
b) ML for consumer protection
We deal with issue of unlawful clauses in Terms of Service online (ToS), that is clauses which directly threaten users' concrete interests for example regulating how data will be managed and the liabilities on such. We experiment the use of ML and advanced language processing techniques to support the classification of ToS clauses categories and fairness level, finding that our techniques overcome state-of-the-art solutions and can be used to measure the ToS unfariness. The conceived approach has been embedded

into *ToSware*, a tool to raise consumers *awareness* against unlawful practices in online ToS. The main features of *ToSware* are: *(i)* having a mechanism to measure the unfairness of online ToS, *(ii)* making ToS more easy to read thanks to different visualization techniques and visual metaphors evaluated by real users, *(iii)* a lightweight implementation which does not impact on the users' devices performance.

The dissertation ends up with considerations about the challenges for ML research in these specific areas, and the future perspectives unfolded by computational law and techno-regulation.

Contents

Chapter 1

Research rationale

Over the last decades, the spread of information and communication technologies (ICT) has triggered a pervasive change at all levels of human society. To date, most of our lives takes place online where algorithms shape and guide our behaviour and the governance of our societies [1]. Whether we like it or not, technologies are increasingly being used to nudge, bias, guide, provoke, control, manipulate and constrain human behaviour. Sometimes this is beneficial, sometimes benign, and others problematic [2].

In this digital society, new regulatory mechanisms are needed to order a "hybrid" reality in which technologies and social activities melt in an inextricable whole. The scenario raises new challenges to legal systems and public authorities: traditional regulations and policies often look unsuitable to safeguard rights in digital settings. Legal safeguards are increasingly bound to be conceived and developed together with the technological ones.

The issue can be traced back to "techno-regulation", an emerging regulatory paradigm that can be in the first place defined as "*the effectuation of norms through technical means at various levels such as rule-making, implementation, monitoring and enforcement*" [3]. Taking hold in all the scenarios where social and economic interactions are mediated by the ICT, techno-regulation confronts us with a perspective in which technology is a subject and an active part of the regulatory process at once. Due to this hybrid nature, techno-regulation raises issues on the legal level and, at the same time, puts challenges on both computer science and computational legal research. The idea of implementing normative strategies through digital technologies, indeed, calls not only for adequate regulatory frames but also for IT solutions properly supporting legal safeguards.

In the following, we will see the perspectives of using technologies for regulative strategies framing techno-regulation (Section 1.1) and then how those can be applied for the case of privacy (Section 1.2; 1.3).

1.1 Code is law

In the introduction, we touched on the difficulties that traditional legal remedies come across when attempting to uphold the law in digital society: the need of a technological support for legal safeguards is growing.

On closer inspection, human artifacts have always contributed to shaping the law in terms of contents, complexity, and ways to protect the rights. The "*offendicula*" known since the ancient Roman law - gates and shards of glass put on the edge of boundary walls - are an *ante-litteram* example of this: a way to protect rights relating to physical goods (an orchard, a house) from threats taking place in the physical world, using physical artifacts. In the digital society, where the goods to be protected are often intangible just like the threats against them, technology retains its role. Contexts that could be cited in this regard are manifold, just think about the use scenarios of robots and drones.

A special place should anyway be acknowledged to the Internet, a universe that is deeply connected with the idea of algorithmic governance [2]. It is no accident that the theoretical reflection about techno-regulation went along with the rise and evolution of the World Wide Web. An indirect confirmation of this relation can be found analyzing the number of scientific publications dealing with techno-regulation over the years. The trend shows how the development of interest in the regulatory use of technology coincides with the emergence of Web 2.0 (see Figure 1.1). Internet is not only the place where the normative role of code has been identified and discussed for the first time, but it has also provided legal scholars and computer scientists with cues about the possibility of purposely using the code, computational tools, and web technologies to implement legal safeguards.

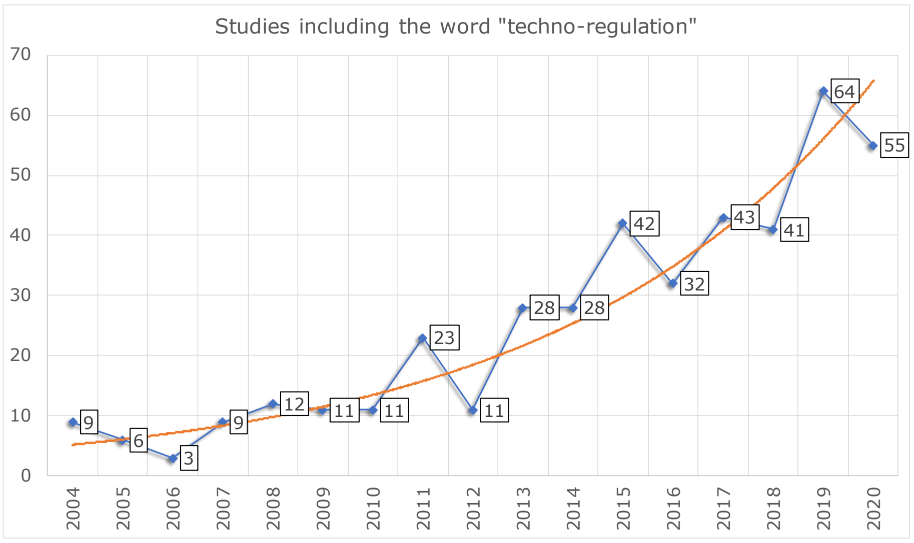


Figure 1.1: Number of papers including the word "techno-regulation" on Google Scholar from 2004 to 2020.

In this respect, it makes sense casting a glance at the contributions coming from those legal scholars that have so far paid more attention to the impact of ICT on the law and its phenomenology. A cursory review can turn useful to better understand the evolution that led to the emergence of current vision of techno-regulation.

The debate started with a group of scholars that firstly realized how, throughout the Internet, technology plays a regulatory role. Back in 1997, Joel Reidenberg [4], director of Fordham Center on Law and Information Policy, highlighted how technical standards and design choices on the web actually impose rules, just like legal rule-making does. Policies - in the narrow sense of protocols but also policies in the broadest sense - are embedded into the design of technological systems, similarly to what happens in "privacy by design" [5]. The vision is taken to the extreme by Lawrence Lessig [6], from Stanford Center for Internet Society, with his claim that in cyberspace "code is law".

Over the years, the way of looking at the potential intersection between law and ICT has turned into the idea of exploiting technology to regulate economic and social relationships both online and offline. The philosopher of law Roger Brownsword was one of the earliest to explicitly state that technological infrastructures supporting our transactions and interactions are a set "*to join law, morals and religion*" as "*one of the main instruments of social control and order*" [7]. A bolder consideration is that of Primavera De Filippi when claiming that, in some ways, technical rules can be even more effective than traditional legal norms [8]. While the latter merely stipulates what people shall or shall not do, the former determine what people can or cannot do in the first place [8]. According to De Filippi, this actually "*eliminates the need for any third party enforcement authority to intervene after the fact, in order to punish those who infringed the law.*"

The transition we are talking about is somehow accomplished with the concept of "code-driven law" recently sketched to identify "self-executing" statutes and contracts directly written into computer code. In the vision proposed by Mireille Hildebrandt [9], code-driven law gives rise to a completely new kind of normativity: "*the upcoming integration of computational law into mainstream legal practice*" is going to "*transform the very nature of existence of law*".

The definition that best summarizes all the facets of techno-regulation is probably the one proposed in [10] that describes it as the "*intentional influencing of individuals' behaviour by building norms into technological devices.*" In this case, the attention is put not only on automated *enforcement* of legal norms but also on the possibility to *nudge* people. The perspective extends the concept of techno-regulation up to include the design of technical solutions aimed to affect behaviors by promoting norm compliance through targeted warnings or suggestions [10, 11]. Ultimately, early forms of ICT-enabled techno-regulation date back to the 2000s with Digital Right Management (DRM), a way to implement copyright law safeguards into technological means by limiting the copy of multimedia contents. Since then, technological development and research at the boundaries between law and computer science led to more sophisticated ways to integrate regulation and IT.

A review of the most interesting proposals and experiments is offered in [12]. It emerges that the techno-regulation paradigm is already producing tools somehow capable of pursuing the same goals normally pursued by means of traditional legal instruments. We are facing an entirely new dimension of legal regulation showing different interesting aspects. A first one lies in the fact that, instead of having to be entrusted with *ex-post* regulations by the legal bodies or the police, techno-rules can be enforced *ex-ante*, being their violation impossible or still very difficult. Moreover, in contrast to traditional legal rules, intrinsically ambiguous and open to interpretation, techno-rules are highly formalized and leave little room for ambiguity, thus reducing the likelihood of misunderstanding and lawsuits. Based on the current debate (see, among the most recent contributions [13, 3, 14, 15]) and on an analysis of experiences made both in private and public contexts, we can sketch a tentative categorization of the directions taken by techno-regulation. It is possible to split into two categories:

- *Fact detection/classification*: techno-rules and tools are firstly aimed at identifying facts and individuals to which particular regulatory and/or legal consequences must be linked. Activities of different types can be traced back to this category: (i) detection of breaches of norms (both private regulation and legal provisions); (ii) identification of individuals deserving protection; (iii) identification of individuals responsible for illegal/prohibited conducts, and so on.

- *Counteract*: the second category includes tools aiming to concretely safeguard interests protected by norms and regulations. This firstly turns into technical solutions materially preventing conducts regarded as harmful or illegal, known as *enforcing* solutions. Then, other softer forms of (intentional) technological influencing can be mentioned. As already emphasized by scholars in the field of *Law and Technology* (see [10, 11, 16]), technologies may also be used to persuade, or to *nudge* individuals promoting *awareness* and the compliance with rules.

1.2 Privacy in the digital society

As we above explained, in the digital society the pervasiveness and advancement of ICT have led to a radical change in the way citizens live, act, and interact, threatening to intrude different human rights. Among these, *privacy protection* has become one of the most debated ones.

The discussion of privacy is, actually, a very old topic renewing every time there is the introduction of a new technology. When photography was first introduced in newspapers, people were concerned about potential privacy violations that could happen due to the publication of photos that violate the privacy of an individual. This problem was so prominent that Samuel D. Warren and Louis D. Brandeis, at the end of the 19th century, introduced one of the most well-known definition of privacy, i.e., "*right to be let alone*" [17], because of the problems arising due to the new technology, enabling what has called "social withdrawal". Later, privacy has also been defined as the *right to prevent the disclosure of personal information* [18]. Privacy definitions are very context-based, as it can be argued from the following one: "*individuals have privacy to the extent that others have limited access to information about them, to the intimacies of their lives, to their thoughts or their bodies*" [19]. Recently, a group of researchers [20] from the Faculty of Law from the University of Haifa, gave the following definition: "*The right to privacy is our right to keep a* *domain around us, which includes all those things that are part of us, such as our body, home, property, thoughts, feelings, secrets and identity. The right to privacy gives us the ability to choose which parts in this domain can be accessed by others, and to control the extent, manner and timing of the use of those parts we choose to disclose.*" More recently, privacy has also been defined as "*having the ability to control the dissemination of sensitive information*" [21]. The American privacy scholar and lawyer Daniel Solove made an insightful attempt to approximate the concept of privacy in terms of six categories that are partly overlapping, while thus covering much of what we intend when referring to privacy: (a) the right to be let alone, (b) limited access to self, (c) secrecy - concealment, (d) control over personal information, (e) personhood - protection of identity, dignity, (f) intimacy.

The main reason why privacy has become one of the crucial concerns for the digital society is the increasing risk for Internet users of what can be coarse-grain defined as personal data leakage. Online browsing, banking, shopping, social network interactions, and any type of online economic, social, personal collaboration and communication could undermine the individuals' privacy due to a variety of factors that include not only the frightening increase of information leakage. We have witnessed to a succession of ever more pervasive technologies, which, in the last decades, have enabled the extraction and gathering of an unprecedented volume of citizens' personal information. They have violated, and still violate, principles of the information security and privacy by unregulated access to information and personal data. The real problem, however, is not the collection itself, but the consequences of such gathering linked to the tremendous computational power and the capability of data analytics when unveiling every trait of one's personality, characteristics, attitudes and life which often touch issues even beyond privacy [2]. Against this backdrop, securing private data, devices and users' privacy in the digital society has become an utmost concern for individuals, business organizations, national governments and researchers.

Adapting the well-known taxonomy proposed by T.

Zarsky [22], we can identify two main problems undermining the privacy of individuals (see an overview in Figure 1.2):

- *lack of consent or comprehension* - the user intentionally switches on applications, software or devices, but it is unaware - for various reasons - of the related risks.
- *unfair or illegal transfer of wealth* - the user is a victim of malicious practices stealthy perpetrated by an "adversary", such as a cyber attack.

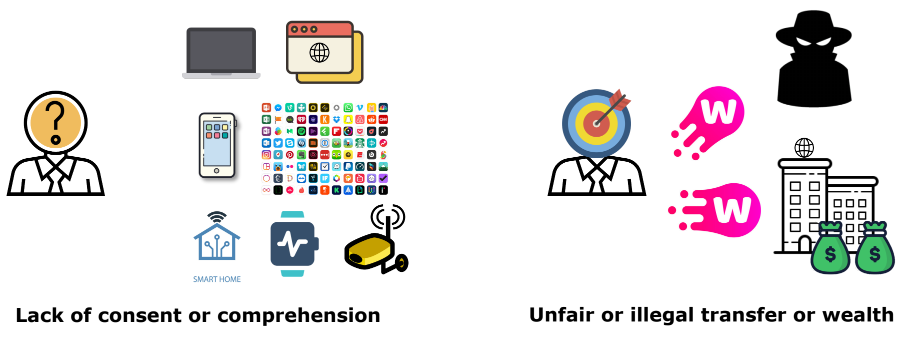

SMART HOME
Lack of consent or comprehension
Unfair or illegal transfer or wealth

Figure 1.2: Problems undermining the privacy of individuals, adapting the taxonomy proposed by T. Zarsky [22].

As for the first point, given the complexity, degree of intertwining, inscrutability and ontogenesis of the ICT we make use daily [2], users feel unarmed and concerned about their privacy [23, 24, 25, 26, 27, 28, 29]. Just think, for instance, of the tough task of fully understand privacy policies, Terms of Service or apps settings on a smartphone. That is why enforcing laws and safeguard rights in new technologies have been guiding principles and consequently large part of the research in the past 15 years has been attracted from issues such as privacy online [30], privacy by design [5], privacy by default [31], and techno-regulation (e.g., [10, 32]) in general (see Section 1.1).

Yet, all new technologies need to deal with malicious practices perpetrated by "adversaries", such as cyber attacks of various forms and with different targets within the individuals' information assets (e.g., health data). In this respect, in the last years, the number of cyber attacks has increased at a high pace: as of 2020, the average cost of a data breach in the United States amounted to 8.64 million U.S. dollars. The global average cost of a data breach in the measured period was 3.86 million U.S. dollars[1]. And a large part of the cyber crimes involves directly individuals[2]. For these reasons, another part of the research has focused on cybersecurity [33], proposing defensive strategies against threats so to securing privacy.

1.3 Protecting the right to privacy

As highlighted above, Section 1.1, during the years, many studies on how to safeguard rights in digital society have been proposed, and the envisioned approaches can be fit for case of privacy right. While legal scholars tended to focus on traditional "command and control" techniques in which the law prohibits specified conducts, backed by coercive sanctions for violation (e.g., GDPR), computational legal scholars and computer scientists have explored how design and code operate as a regulatory instruments [34, 35, 36]. Solutions explored heading in this direction can be traced back, according to recent analysis [13, 32], to two fundamental approaches. The first group of solutions includes tools designed to concretely safeguard interests protected by norms through technical solutions directly producing effects defined by legislators. Smart contracts are a prime example of this approach: they are self-executing and self-enforcing computer programs capable of automatically connecting legal and economic effects (e.g., a refund of travel expenses) to the occurring of given facts (e.g., the train delay). As such, they "determine what people can or cannot do in the first place" [8].

[1] `https://www.statista.com/statistics/463714/cost-data-breach-country/`

[2] `https://www.statista.com/statistics/1085784/cyber-attack-share-global-by-category/`

The second group of solutions turns into "softer forms" of intentional technological influencing of human behavior aimed not at enforcing but at promoting norm compliance. This approach largely draws on insights from nudge theory [37], a research perspective fed by behavioral economics and psychology, putting forward the idea that positive reinforcements and indirect suggestions, leveraging cognitive features and biases of human being, can influence the behavior and decision making of groups or individuals often outperforming other ways to achieve compliance, such as education, legislation or enforcement. Rapidly become a reference paradigm for administrators and policy makers[3], the nudge perspective has also seeped into the techno-regulation (see [10]), inspiring new ways to use technology to influence individuals' behaviors via targeted warnings, messages and suggestions promoting awareness and "gently" nudging decision making. According to the schema above sketched, we can distinguish between two ways of effectively protecting the privacy right, i.e. *enforcement* (see Section 1.3.1) and *nudge-based* (see Section 1.3.2) strategies.

1.3.1 Enforcement strategies hampering breaches of law

Enforcement solutions aim at hampering breaches of law and impose constraints. They materially prevent conducts regarded as harmful or illegal by directly protecting users. Solutions of this kind act as first (e.g., the user's Web browser blocks dangerous contents), third (e.g., an external user software blocks dangerous contents on the Web) or fourth parties (e.g., the network used blocks dangerous contents on the Web) often without requiring the user intervention or decision.

[3] David Cameron set up a *Behavioural Insights Team* (BIT or Nudge Unit) at the centre of UK government in 2010. See more at `https://www.bi.team`

1.3.2 Nudge-based strategies raising awareness

Nudge-based solutions are a softer form of regulation [16, 10]. The solutions belonging to this category aim to protect privacy by promoting awareness in the users' behavior. In this case, users have to take the decisions needed to uphold privacy rights. This happens basically in two ways. The first one, less "enhanced", shows users *static nudges* - to use a well-known metaphor, such as salad in front of the fried chicken to encourage healthy eating - which consist in standard messages. The second one, also referred as *"Hypernudge"* [38], provides users with continuously updated, tailored warnings that shape the informational choice context in which individual decision-making occurs, taking into account the results of behavioral analysis.

Therefore, in this research work, we consider a "*protection* of users' privacy" any technological solution which *enforces* constraints without the users intervention or *nudges* users letting them to make the relevant decision.

In Figure 1.3, the proposed reading key for the privacy issues and corresponding protections it is shown. It is possible to grasp the mapping between the problems highlighted in Section 1.2 and the existing protections to effectively safeguard the privacy right.

1.4 Research Overview

In the light of the above, a number of safeguarding technical solutions to uphold privacy legal standards have been proposed, but those based on static mechanisms have often failed due to the hybrid and complex reality to deal with. Just to make an example, think about Do Not Track System or blacklisting mechanisms for trackers, malware or phishing [39, 40, 41]. Thus, it emerged the need for not static solutions suitable to the ever-evolving threats. In this respect, *machine learning* (ML), with its pattern recognition capability, appears to be a precious ally to ensure not only

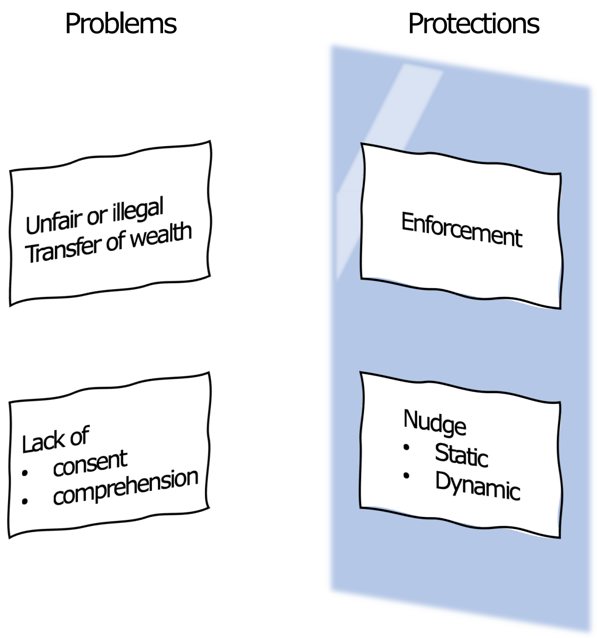


Figure 1.3: Our new reading key for the privacy problems and corresponding protections. Problems in the map are an adaptation of the taxonomy proposed in [22].

users' privacy but also other rights threatened in the digital society.
Focus. In this dissertation, we *focus* on intelligent *privacy* safeguards for users in the digital society.
Objective. Our *goal* is to explore, both on the theoretical and experimental levels, how ML approaches can be fit to support the protection of privacy and the related rights. The research draws also upon most recent development in the areas of computational law and techno-regulation, two research paradigms emerging, at planetary scale, on the boundaries between computer science and law.
Structure. This dissertation includes the following chapters:

- Chapter 2 depicts the theoretical and methodological framework of this work, through a systematic literature review framing the use of ML to protect users' privacy. In doing this, we trace back existing approaches and solutions for the privacy protections to two fundamental categories: *enforcement* (i.e., solutions which *impose constrains* and hamper breaches of norms) and *nudge* solutions (i.e., solutions which inform users and increase their *awareness* to promote privacy-oriented behaviors). We provide a comprehensive taxonomy of main areas, threats, ML methods, type of protection delivered analyzing 143 studies published from January 2017 to October 2020. This represents an original contributions to be submitted to a top quality journal.

- Chapter 3 depicts the research design adopted for the activities presented in the following chapters;

- Chapter 4 presents ML for privacy enforcement. In this chapter, we deal with the long-standing issue of third-party tracking on the Web in which users' private data are unfairly stolen for marketing and malicious activities, such as online stalking. We experiment the use of ML to distinguish between trackers and functional resources on the Web, finding that such techniques can be fit for a high classification rate of such threat. The resulting ML-based approach has been implemented into *GuardOne*, a tool to protect users against third-party tracking, which provides *enforcement* solutions to block trackers. *GuardOne* has been evaluated in real-world against similar commercial privacy enforcing solution. The main features of *GuardOne* are: *(i)* a hybrid mechanism based on ML and blacklisting, *(ii)* customization based on the user browsing habits, *(iii)* a very lightweight implementation which does not impact on the users' devices performance compared to commercial solutions, *(iv)* a high effectiveness in detecting and blocking third-party trackers better than the vast majority of commercial solutions. The results have been published through the proceedings of *International Conference on Information Security and Privacy 2020* and *Computer Networks, Elsevier*.

- Chapter 5 presents ML for privacy awareness. In this chapter, we deal with the issue of unaware and/or uncontrolled dissemination of personal and private data, in text format, on the Internet. We experiment the use of ML to support both the classification of the text topic (among the most sensitive ones, e.g., politics and health) and the sensitiveness of the content according to such topic, finding that it is easier to classify longer texts than short messages. Furthermore, we experiment how the ML solutions designed can be fit to learn the user's personal attitudes towards privacy. We then embed such ML-approaches into *Knoxly*, a tool to protect users against the dissemination of personal and/or private information online, which relies on *nudge*-based solutions. Specifically, *Knoxly* aims to raise awareness and promote privacy-oriented behavior by means of alerts/warnings. The main features of *Knoxly* are: *(i)* a Keyword module to detect common sensitive words and personal identifiable information, *(ii)* a Topic module to distinguish the text's topic, *(iii)* a Sensitiveness module to "measure" the sensitiveness of the text content, *(iv)* a Customized module allowing the user to personalize the warnings displayed, *(v)* an intuitive User Interface powered by Visual Analytics techniques, *(vi)* a lightweight implementation which does not impact on the users' devices performance. This represents an original contributions to be submitted to a top quality journal.

- Chapter 6 expands the research scope presenting ML for *child protection*. In this chapter, we deal with the challenge of providing online (privacy) protections for a specific category of users, that is children, which according to the General Data Protection Regulation (GDPR) and UNICEF need *ad-hoc* safeguards. Within this research project, named *AI4Children*, we experiment several ML-based approaches for users identification which can be seen as the baseline to uphold legal standards for online child protection. In fact, once identified the user, it is possible to trigger the specific safeguards. In more details, the conceived approach (based on data integration techniques) is capable of recognizing the age of a user based on the touch gestures he/her performs on a mobile device with a high accuracy. The main features of *AI4Children* are: *(i)* distinguishing between adults and underages based on commonly performed touch gestures, *(ii)* using a small set of features and touch gesture to perform a high accurate classification, *(iii)* robustness in the classification on different devices. The result has been published through *Multimedia Tools and Applications, Springer*.

- Chapter 7 expands the research scope presenting ML for *consumer protection*. In this chapter, we deal with issue of unlawful clauses in Terms of Service online (ToS), that is clauses which directly threaten users' concrete interests for example regulating how data will be managed and the liabilities on such. We experiment the use of ML and advanced language processing techniques to support the classification of ToS clauses categories and fairness level, finding that our techniques overcome state-of-the-art solutions and can be used to measure the ToS unfariness. The conceived approach has been embedded into *ToSware*, a tool to raise consumers *awareness* against unlawful practices in online ToS. The main features of *ToSware* are: *(i)* having a mechanism to measure the unfairness of online ToS, *(ii)* making ToS more easy to read thanks to different visualization techniques and visual metaphors highly rated by users, *(iii)* a lightweight implementation which does not impact on the users' devices performance. This represents an original contributions to be submitted to a top quality journal.

- Chapter 8 conclude with considerations about the challenges for (ML) research in these specific areas, and the future perspectives unfolded by computational law and techno-regulation.

Chapter 2

Building research background: a systematic literature review on the use of ML to protect users' privacy

Chapter Highlights

- Review of 148 studies published between 2017 and 2020;
- Tracing-back the delivered solutions of reviewed studies to two categories, i.e., enforcement and nudge;
- The big picture on the use of ML to protect users' privacy;
- A comprehensive taxonomy of reviewed studies including the main ICT areas, threats, ML methods, type of protection delivered;
- Web is the most covered area, the most cited area is instead Mobile, the most faced threats are those related to Data leakage, the most used ML methods are neural networks, and the Technological Readiness Levels of reviewed works is relatively low.

2.1 Introduction

Recently, a load of studies has leveraged ML as a strategy to address the privacy issues in our digital society, therefore we have found several surveys. Such surveys and reviews specifically focused on two main topics, i.e., cybersecurity, and Internet of Things (IoT) with a few others which tackled privacy in Online Social Networks (OSNs) and Android platforms.

As for the case of cybersecurity, in [42], the authors identified the types of phishing attacks and current methods used in preventing them, finding that ML methods are widely used in the area. In [43], authors describe a focused literature survey of ML and data mining methods for cyber-analytics in support of intrusion detection. They mainly addressed the complexity of ML approaches, discussing the challenges for using ML for cybersecurity. In [44], the authors reviewed the intersection of artificial intelligence (AI) and cybersecurity, summarizing existing research efforts in terms of combating cyber attacks using AI, including ML solutions, and analyzing the counterattacks from which AI itself suffer. Recently, in [45], the authors provide a comprehensive survey of the works that have been carried out from 2013 to 2018 on ML in cybersecurity, describing the basics of cyber attacks and corresponding defenses, the basics of the most commonly used ML algorithms, and proposed ML and data mining schemes for cybersecurity in terms of features, dimensionality reduction, and classification/detection techniques. In this context, authors also provides an overview of adversarial ML, including the security characteristics of deep learning methods.

As for the IoT context, in [46], the authors investigated the attack models for IoT systems and reviewed the IoT security solutions based on ML. ML-based IoT authentication, access control, secure offloading, and malware detection schemes to protect data privacy are the focus of the work. Furthermore, they discuss the challenges that need to be addressed to implement these ML-based security schemes in practical IoT systems. In [47], the authors presented a holistic picture of IoT, fog computing, and ML techniques for securing IoT devices and fog computing systems. Their attention has been put on ML techniques for detecting abnormalities and attacks. In [48], the authors provide an overview of privacy preservation techniques and solutions proposed in literature along with the IoT levels at which privacy is addressed by each solution as well as their robustness to privacy breaching attacks. Their work analyzed functional and non-functional limitations of each solution, surveying on ML applications designed with these solutions. It ends up identifying open issues in such solutions applied to IoT environments, especially regarding compliance with General Data Protection Regulation (GDPR). More recently, in [49], the authors review studies leveraging ML as a strategy to address the privacy issues of IoT, including scalability, interoperability, and resource limitation. The survey examine opportunities and concerns related to utilizing data in ML-based solutions for privacy in IoT. It explores different data sources in IoT and categorize them, and examine the extent to which some data categories have been used with ML-based solutions to preserve privacy. Lastly, in [50], authors systematically reviewed the security requirements, attack vectors, and the current security solutions for the IoT networks. They then shed light on the gaps in these security solutions that call for ML and deep learning approaches.

About OSNs, in [51], authors focus on the pervasiveness and privacy risks of OSNs. They provide an overview of the privacy and security issues that emerged so far in OSNs, introducing a taxonomy of privacy and security attacks in OSNs. Lastly, the authors overview existing solutions - some of which are based on ML - to mitigate those attacks and outline challenges still to overcome.

Lastly, concerning Android platforms, in [52], authors survey security solutions that make use of ML approaches to detect malware in Android, aiming to provide the best possible approach. They analyzed the architecture of these approaches and present the taxonomy of Android OS based security solutions.

There are also other surveys which may appear to have contact points with this paper, e.g., [53] where authors studied and reviewed how to protect ML solutions from attacks trying to unveil private data they may use. Actually, this is a different topic because we point at strategies which make use of ML to directly protect users' privacy.

Analyzing all the articles found in this research, no one proposed a systematic literature review aiming to identify different key areas of research on the use of ML to protect users' privacy. All these works have a specific area of interest to focus on when considering privacy solutions. Instead, this work aims to offer a big picture of the solutions to protect users' privacy regardless of the research area. Furthermore, we trace back existing approaches and solutions for the privacy protections to two fundamental categories: *enforcement* (i.e., solutions which impose constrains and hamper breaches of norms) and *nudge* solutions (i.e., solutions which inform users and increase their awareness to promote privacy-oriented behaviors). According to [54], a systematic review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question or topic area. A work of this nature is essential to privacy and security research area because it frames threats, involved methods, software architectures, challenges and open questions, and the opportunities regarding the use of ML to protect users' privacy. However, as pointed out in other works [55, 56] the systematic literature reviews have some limitations due to the empowered methodological rigor. Furthermore, the current work is limited to researches that use ML to protect users' privacy in the digital society.

Contributions In this review, the following *main points* are covered:

- An overview of how to protect privacy is briefly highlighted and analyzed.

- A new reading key for the research in the field of ML to protect users' privacy is provided tracing back existing approaches and solutions to the categories of *enforcement* and *nudge.*

- A big picture about the current use of ML to protect users' privacy is offered.
- A taxonomy to classify the areas, threats, ML methods, type of protection delivered is depicted.
- Inferences and recommendations are given.

Chapter structure The rest of the paper is organized as follows: Section 2.2 illustrates the main ML methods adopted in literature; Section 2.3 details the methodology employed to perform this systematic literature review; Section 2.4 presents the results obtained from the collection of articles and discuss the main findings; Section 2.5 presents the limitation of this work; finally, we discuss the conclusions and directions for future work in Section 2.6.

2.2 Machine learning in a nutshell

ML is a research field that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [57]. It has been used for a variety of applications (e.g., activity recognition [58], stock price prediction [59], healthcare [60], business [61], autonomous vehicles [62], etc.), especially in the last 20 years when large datasets and powerful machine have become more common. For the process of learning (model fitting) the ML algorithms (also known as models or methods) need to have available some observations or data (also known as samples or examples) in order to explore potential underlying patterns, hidden in the data. These learned patterns are nothing more that some functions or decision boundaries.

A ML approach usually consists of two phases: training and testing. Often, the following steps are performed:

- Identify class attributes (features) and classes from training data.
- Identify a subset of the attributes necessary for classification (i.e., dimensionality reduction).
- Learn the model using training data.
- Use the trained model to classify the unknown data.

Actually, for most ML approaches, there should be three phases: *training*, *validation*, and *testing*. ML methods often have hyper-parameters such as the number of layers and nodes for a Neural Network. To decide which one to use and have a good estimation of the error it will achieve on a test set, there should be a third separate data set, the *validation* data set. The model that performs the best on the validation data should be the model used, and should not be fine-tuned depending on its accuracy on the test data set. Otherwise, the accuracy reported is optimistic and might not reflect the accuracy that would be obtained on another test set similar to but slightly different from the existing test set. For this reason, it is usually performed a task named k-fold cross-validation, a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. When a specific value for k is chosen, the procedure becomes 10-fold cross-validation.

ML algorithms are usually categorized as *supervised* or *unsupervised* with some cases at the boundaries between the two categories (Figure 2.1).

Supervised learning models These models need to process a dataset with some observations and the related labels. For example, the observations could be the software source codes. Such models learn from the labeled dataset and then are used to predict new and unseen samples. For the training procedure, the input is a known training dataset with its corresponding labels, and the

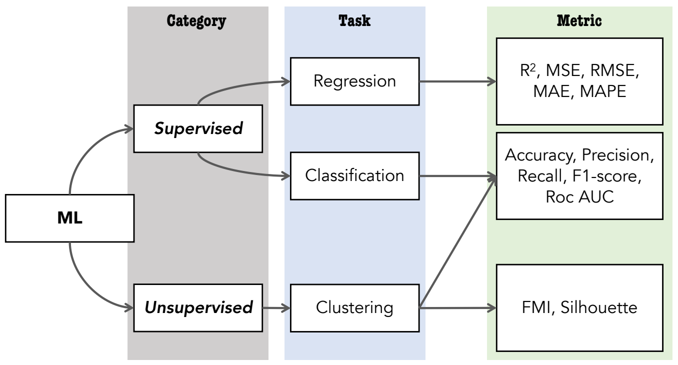


Figure 2.1: A taxonomy of ML methods' *categories* and *metrics* (most) used for measuring performance against different *tasks*.

learning algorithm produces an inferred function to finally make predictions about some new unseen observations that one can give to the model. The model is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct intended output (ground truth label) and find errors in order to modify itself accordingly (e.g., via back-propagation [63]). Supervised models can be further grouped into *regression* and *classification* based on the specific task they are used to address:

- Classification: it is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y), e.g., classifying between trackers and functional resources on the Web [64]. The output variables are often called labels or categories.

- Regression: it is the task of approximating a mapping function (f) from input variables (X) to continuous output variables (y), e.g., stock price prediction [59].

Some examples of models that belong to this family are the following: Support Vector Machines [65] (SVM), Neural Networks like MultiLayer Perceptron [63] (MLP), Convolutional Neural Networks (CNN), Recurrent Neural Network such as Long Short-Term

Memory [66, 67, 68] (LSTM), Ridge Regression [69] (Ridge), random forests [70] (RF), Adaptive Boosting [71] (AdaBoost), K-Nearest Neighbors [72] (KNN), and so on.

Unsupervised learning models These models require to have at hand a dataset with some observations without the need of having also the related labels/classes. Unsupervised learning studies how systems can infer a function (f) to describe a hidden structure from unlabeled data. Unsupervised models we describe are focused on *clustering*, i.e., the task aiming to unveil the inherent groupings in the data, such as grouping OSNs users based on some characteristics/features, e.g., the text messages they disseminate [73].

Some examples of models that belong to this family are the following: Principal Component Analysis [74] (PCA), K-means [75], DBSCAN [76], mixture models [77], the more complex AutoEncoders [78, 79, 80] and so on.

Metrics for evaluating ML methods performance There are several classification, regression and clustering metrics for ML methods. Certain metrics are called by two or even three different names. In the Appendix .1, the identified studies of this review are described and results reported with the metrics and values of such. To understand that section easier, the metrics are described next.

Given the target value (known label) y_i, and the model's predicted value (predicted label) $\hat{y}_i$, and N samples, the metrics used for a *regression* task are:

- *Mean Squared Error* (MSE): $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. MSE measures the square of differences between predictions and target values and computes the mean of them.

- *Root Mean Squared Error* (RMSE): $RMSE = \sqrt{MSE}$. RMSE is the square root of the mean of the square of all of the error. The use of RMSE is very common, and it is considered an excellent general-purpose error metric for numerical predictions.

- *Mean Absolute Error* (MAE): $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$. MAE is an arithmetic average of the absolute errors. MAE uses the same scale as the data being measured. This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series using different scales.

- *Mean Abslute Percentage Error* (MAPE): $\frac{1}{N}\sum_{i=1}^{N}|\frac{y_i-\hat{y}_i}{y_i}|$. MAPE is the MAE expressed in percentage. It works best if there are no extremes to the data (and no zeros).

For all the aforementioned metrics (MSE, RMSE, MAE, MAPE) the higher the value the worse the prediction, therefore a value close to 0 involves an optimal model. There are further metrics like R-squared measures (R^2), i.e., the proportion of the variation in the dependent variable (Y, i.e., the classes) explained by the independent variables (X, i.e., the features). So, if R^2 is 0.8, it means 80% of the variation in the dependent variable is explained by the independent variables. The higher the R^2, the more variation is explained by the input variables and hence better is the model. However, the problem with R^2 is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. Adjusted R-squared ($\bar{R}^2$) adjusts the statistic based on the number of independent variables in the model.

For a binary classification problem, the metrics are computed from the confusion matrix (see Table 2.1).

The metrics frequently used for binary *classification* problems are:

- *Accuracy* or Proportion Correct: $\frac{(TP+TN)}{(TP+TN+FP+FN)}$. Informally, accuracy is the fraction of predictions our model got right. When classes are balanced, this is a good measure;

	Actual class: c_1	**Actual class: not** c_1
Predicted class: c_1	True positive (TP)	False positive (FP)
Predicted class: not c_1	False negative (FN)	False positive (FP)

Table 2.1: Confusion matrix in a binary classification problem.

however, when classes are unbalanced (e.g., 92% of items belong to class c_1 and 8% to class c_2, if all the items are classified as c_1, the accuracy would be 92% but all items from class c_2 would be misclassified), this metric is not very useful.

- Positive Predictive Value (PPV) or *Precision*: $\frac{TP}{(TP+FP)}$. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
- Sensitivity or *Recall* or True Positive Rate or Probability of Detection (PD) or Detection Rate: $\frac{TP}{(TP+FN)}$. Intuitively, recall is the ability of the classifier to find all the positive samples.
- *F1-score* or F-score or F1: $2 * \frac{(Precision*Recall)}{(Precision+Recall)}$. It is the harmonic mean of precision and recall.
- *Roc AUC* score: the most common way evaluate the performance of a binary classifier, is created by plotting True Positive Rate against False Positive Rate.

All such metrics ranges between 0 and 1, with 1 meaning an optimal value.
For a multi-class problem (classification into more than two classes), usually the following metrics are used: *(i)* overall accuracy: exemplars classified correctly, all exemplars; *(ii)* weighted F1-score: weight the F1-score of each class by the number of samples from that class.

For what concerns *clustering* tasks there are several metrics in addition to those foreseen for the classification task. Among these, the most common are:

- The Fowlkes-Mallows index (FMI): a metric that can be used when the ground truth class assignments of the samples is known. The FMI is defined as the geometric mean of the pairwise precision and recall: $\frac{TP}{\sqrt{(TP+FP)*(TP+FN)}}$, where TP is the number of True Positive (in this case, the number of pair of points that belong to the same clusters in both the true labels and the predicted labels), FP is the number of False Positive (in this case, the number of pair of points that belong to the same clusters in the true labels and not in the predicted labels) and FN is the number of False Negative (in this case, the number of pair of points that belongs in the same clusters in the predicted labels and not in the true labels).

- Silhouette Coefficient (Silhouette): a metric used when the ground truth label are not known. A higher Silhouette score relates to a model with better defined clusters. The Silhouette is defined for each sample and is composed of two scores:

  - a, the mean distance between a sample and all other points in the same class;
  - b, the mean distance between a sample and all other points in the *next nearest cluster*.

  The Silhouette s for a single sample is then given as: $\frac{b-a}{max(a,b)}$.

Actually, with regard to the identified studies, we have not found works using specific clustering metrics.

2.3 Methodology

The current study presents a systematic literature review designed to provide a comprehensive overview of the usage of ML to protect users' privacy, summarizing the technologies and methods used, objectives, and identifying promising directions, with in-depth analysis and synthesis. With this in mind, we followed widely recognized guidelines [81, 82, 83, 56] to plan and run systematic mapping studies. The presented systematic literature review method was carried out through the following activities:

1. Planning the review;
2. Investigation of research questions;
3. Description of both sources of information and strategy used to collect data;
4. Definition of the selection and exclusion criteria used to filter the studies;
5. Description of the quality assessment of the selected studies;
6. Comparison of selected studies and research questions.

The following paragraphs describe how this process of mapping the study was carried out.

Planning The review protocol includes the research questions, sources of information, and the methods used to map the studies tackling the use of ML to protect users' privacy. Also, the review comprises the identification of primary studies, applying selection and exclusion criteria and synthesizing the results. In order to reduce researcher bias, one of the authors developed the protocol, the others reviewed and then the authors developed a discussion, review, and iteration. Finally, the authors searched in the databases and reported the results.

Research questions Once the definition of research questions is the most crucial part of any systematic review [83], we aim to identify and classify the existing literature focusing on the use of ML to protect users' privacy as well as the emerging threats, ML solutions, research trends, and the limitations. Table 2.2 describes our research questions.

Identifier	**Research Question**
RQ1	What is the current state-of-the-art for ML-based solutions to protect users' privacy?
SQ1.1	Which application areas are the more supported by ML-based safeguards?
RQ2	What are the main threats to users' privacy?
RQ3	Which ML methods are the most utilized to protect users' privacy?
SQ3.1	What kind of protection do such ML methods deliver?
RQ4	Which research gaps and trends for future research are observable in the literature?

Table 2.2: Research questions guiding this work.

Sources of information and strategy The literature search was conducted between 1 October 2020 and 15 October 2020. The standard search string used to identify literature was:

(*"privacy"*) **AND** (*"computational intelligence"* **OR** *"artificial intelligence"* **OR** *"machine learning"*).

We bounded the search to those works published between January 2017 and October 2020. There were some differences in the search strings used between databases, due to different query languages and limitations between scientific paper databases. The databases queried are shown in Table 2.3 with their corresponding number of results; the results are shown prior to the exclusion of duplicates and prior to their evaluation based on selection and exclusion criteria.

Database Name	**URL**	**Total results**
Google Scholar	`https://scholar.google.com`	16,100
Scopus	`https://www.scopus.com`	4,043
ACM Digital Library	`https://dl.acm.org`	6,965
IEEEXplore	`https://ieeexplore.ieee.org`	1,123

Table 2.3: Databases with search results.

The abstracts of the studies were all inspected to include/exclude the studies from the review. There were some necessary modifications to search strings. All the results were inspected on all databases except for Google Scholar, where the results were shown by *relevancy*, and the search was stopped once there were no more included (relevant) studies on two successful pages (as proposed in [84]). We obtained 21 results, and this criteria was satisfied after 31 Google Scholar pages.

Selection and exclusion criteria After obtaining the studies, we removed the impurities from the search results. As impurities, we mean the names of conferences correlated to the search keywords that were in the search results because of the characteristics of the different electronic databases. Next, we started the selection and exclusion process to filter the works that were not relevant and keep those that were the most representative. In Table 2.4 we specify the *selection* and *exclusion* criteria adopted for this review.

Selection criteria
Article has been peer reviewed
Article is about protecting users's privacy by using machine learning
Exclusion criteria
Article is written in a language different than English
Article is not accessible with authors' university subscriptions
Article only speculates on the possibilities of using machine learning to protect privacy

Table 2.4: *Selection* and *exclusion* criteria for the results found in the databases.

Quality assessment After using the selection and exclusion criteria to select relevant articles that are about the use of ML to protect users' privacy, we performed a quality assessment of the remaining articles. We used as criteria to evaluate the selected articles the purpose of research, contextualization, literature review, related work, and methodology besides the conclusion and results. To reduce the empirical barriers of full-text filtering, we used a set of questions listed in Table 2.5 proposed by [56] that validate if the selected articles met the quality criteria are listed.

Identifier	Full-text question
C1	Does the article clearly show the purpose of the research?
C2	Does the article adequately describe the literature review, background, or context?
C3	Does the article present the related work concerning the main contribution?
C4	Does the article have an architecture proposal or research methodology described?
C5	Does the article have research results?
C6	Does the article present a conclusion related to the research objectives?
C7	Does the article recommend future works, improvements, or further studies?

Table 2.5: The set of questions for full-text filtering proposed by [56].

Works which not meet the quality criteria have been removed. All the database results were further checked for duplicates, and after removal, 143 results remained.

To ensure the consistency of paper processing, the first author read all papers in the final dataset and recorded the paper's key content in a mind map. All authors continuously reviewed, discussed, and updated the mind map. Additionally, we maintained a spreadsheet to record the key features of each paper (task, methods, improvements, dataset, results, etc.).

In Figure 2.2 we summarize the review process.

2.4 Results

In this section, we present the results obtained from the collection of articles from four databases related to the research topic. We aim to answer the proposed research questions in Table 2.2 about the use of ML to protect users' privacy. Following, we describe all the results obtained.

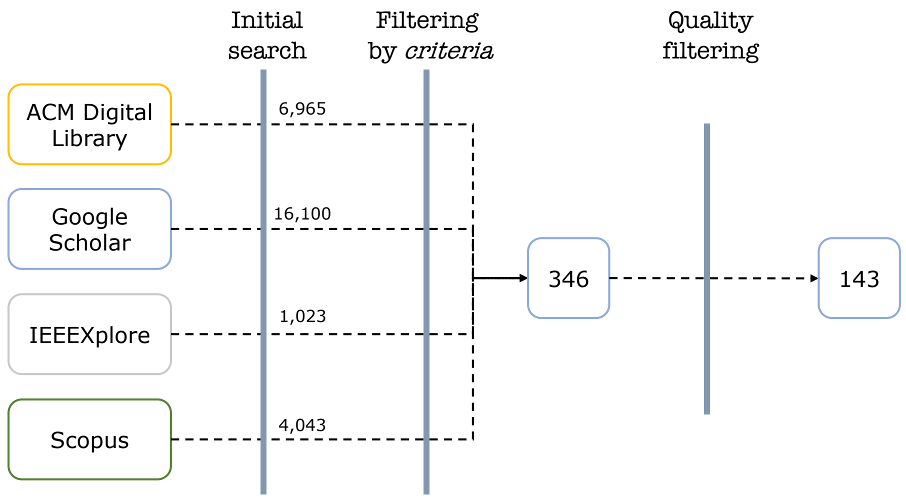


Figure 2.2: Systematic mapping study-article selection.

2.4.1 RQ1: What is the current state-of-the-art for ML-based solutions to protect users' privacy?

The objective of this section is to grasp at a glance what is the current "landscape" for the use of ML to protect the users' privacy by having a coarse-grained overview of the surveyed studies. In particular, we also want to find the application areas more supported by the ML-based safeguards (SQ1.1 Table 2.2). For what concerns the specific ICT areas covered, we have identified studies in the areas of Blockchain, IoT, Mobile, OSN, Web, Windows, and General (when unspecified or covering all areas).

Specifically, Figure 2.3 also shows that the majority of works are in the Mobile area (50 works, roughly 34% of total), followed by Web (38 works, roughly 26% of total). This is due to the change in user habits happened in the last 10-15 years with the advent of smartphones and apps that have almost completely replaced web-based applications.

For an overview on the publication trend (overall and specific areas) over 2017, 2018, 2019, and 2020 we refer the reader to Fig-

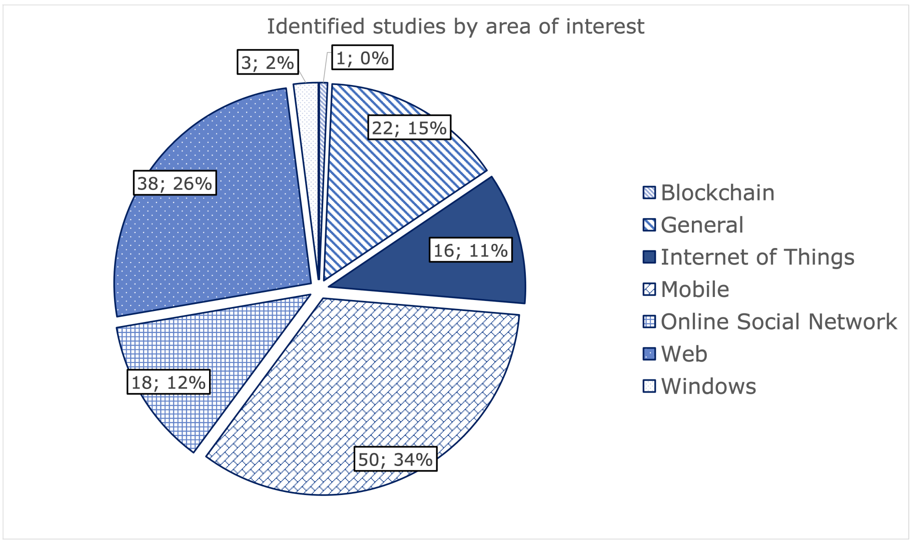


Figure 2.3: Identified studies split by the ICT area of interest.

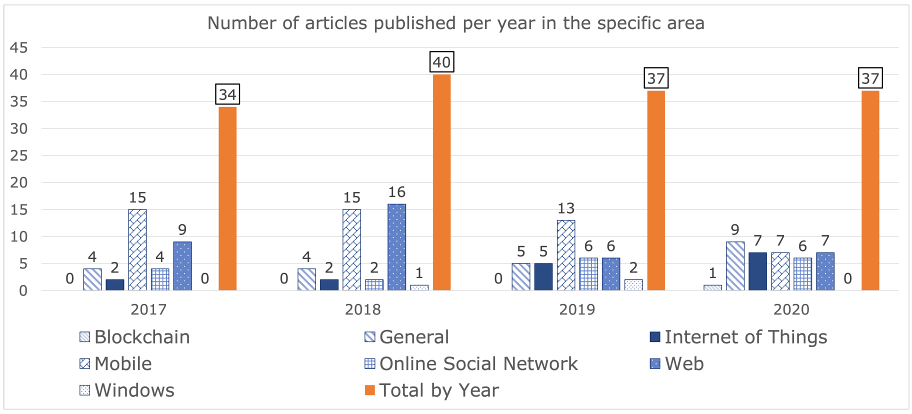


Figure 2.4: Number of articles published per year in the specific ICT area.

ure 2.4. Overall, the bar plot shows that there is a very subtle difference between the 4 years analyzed, with a number of published papers about the use of ML to protect users' privacy which consistently sit around 37 per year (maximum obtained in 2018 with 40 works, and minimum obtained in 2017 with 34 works).

The year 2018 shows the highest peaks of identified works in the areas of Web, 16 studies, and Mobile, 15 studies (for this last area on par with what found for 2017).

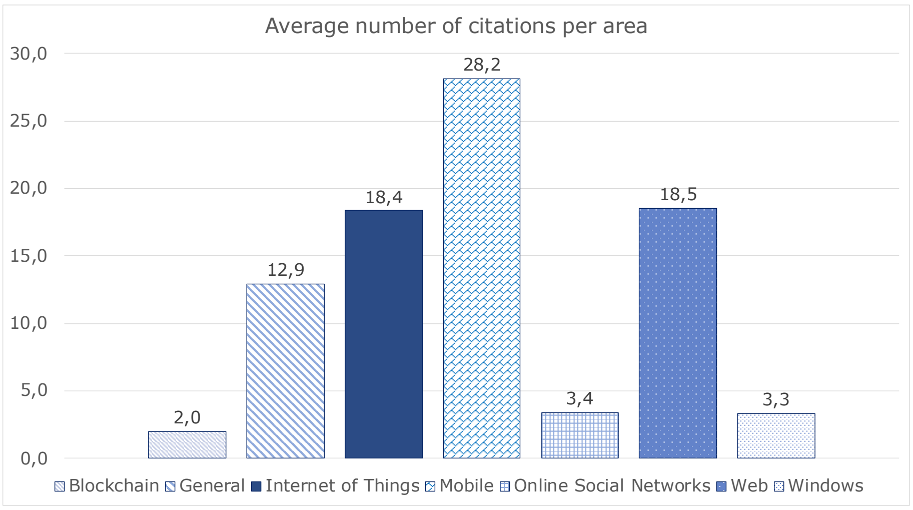


Figure 2.5: Average number of citations for identified articles split by ICT area of interest. Citations have been crawled via Google Scholar.

In addition, we can see that Mobile area occupies the first place in the top three of citations (Figure 2.5) with an average of 28.2 citations per article, followed by Web with an average of 18.5 citations per article and IoT as third place with an average of 18.4 citations per article. Even if the majority of identified studies is not in the IoT area, we understand that the those we reviewed have been very popular in recent years given the high average number of citations.

Lastly, we summarize all the works in Table 2.6, and for the sake of clarity, in Appendix .1, we show the most important studies[1] split into area of interest, thus we have Table 1 devoted to detail studies found in the Blockchain area, Table 2 devoted to

[1]The importance of a work/study has been measured leveraging on Scimago ranking for journal papers and on the Conference Rank for conference paper. In particular, we selected journals paper whose Scimago ranking places

detail studies found in the OSN area, Table 4 devoted to detail studies found in the IoT area, Table 3 devoted to detail studies found in the Mobile area, Table 5 devoted to detail studies found in the Web area, and Table 6 devoted to detail studies found in the broader area of privacy in general. For such (important) studies we show (i) the type of work (*Type*) which could be either "Study" or "Tool" or "Framework", (ii) the *Threat* faced, (iii) the *Highlights* of the work, and (iv) the type of protection provided (see Section **??**).

Area	**Identified studies**
Blockchain	[85]
General	[86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107]
IoT	[108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123]
Mobile	[124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173]
OSN	[174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 21, 184, 185, 186, 187, 188, 189, 190]
Web	[191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228]
Windows	[229, 230, 231]

Table 2.6: Identified studies split by ICT area of interest.

2.4.2 RQ2: What are the main threats to users' privacy?

The objective of this section is to highlight the main threats for the user's privacy faced in the surveyed studies. The identified studies

in Quartiles ≥ Q2 and paper submitted to conferences whose rank ≤ 500. The Scimago ranking is available via https://www.scimagojr.com, while the Conference Rank has been obtained via https://www.guide2research.com/topconf/.

split by threat faced are available in Table 2.7. We have identified the following threats: Anomalous Users, Cyber Attack, Disclosure, General leakage, Health leakage, Image leakage, Location leakage, Malware Windows, Malware Android, Malware (Other), Phishing, Privacy Policy, Ransomware, Spam, Speech leakage, Sybil Attack.

Threat	**Identified studies**
Anomalous Users	[184, 189]
Cyber Attack (general)	[118, 122, 85, 109, 105, 96, 120] [110, 114, 209, 92, 211, 213, 94, 140, 144, 223]
Disclosure	[174, 175, 178, 182, 113, 99, 21, 187, 179, 190, 180]
General leakage	[111, 108, 126, 124, 136, 152, 117, 155, 93, 214, 103, 219, 172] [131, 197, 130, 90, 139, 207, 107, 147, 157, 166]
Health leakage	[89, 115, 106]
Image leakage	[177, 91, 181, 183] [97, 176, 151, 186, 100]
Location leakage	[87, 188, 104]
Malware Windows	[229, 230, 231]
Malware Android	[128, 133, 137, 138, 165, 163, 170, 169, 156, 112, 123, 158, 160, 168, 119, 161, 162] [171, 125, 129, 150, 146, 132, 135, 134, 154, 153, 141, 142, 148, 143, 149, 145, 167]
Malware (Other)	[95, 121] [191, 192, 127, 194, 199, 200, 205, 206, 203, 202, 204, 215, 208, 224, 216, 212, 210, 226, 227]
Phishing	[195, 193, 196, 201, 228, 217] [222]
Privacy Policy (PP)	[86, 225, 198, 164, 218] [173, 220, 221]
Ransomware	[98]
Spam	[101] [159]
Speech leakage	[116, 102] [88]
Sybil Attack	[185]

Table 2.7: Identified studies split by threat faced.

In addition, we present a taxonomy of such threats comprising of four main groups, i.e., *Data leakage*, *Malware*, *Cyber attack* and *Others*. The taxonomy is made as follows:

- in *Data leakage* we have: General leakage, Image leakage, Health leakage, Speech leakage, Location leakage, and Disclosure;

- in *Malware* we have: Android malware, Windows malware, Ransomware and Other type of malware;
- in *Cyber attack* we have: Phishing, Sybil Attack, and General Cyber Attack;
- in *Others* we have: Privacy Policy, Anomalous users, and Email Spam.

As it is possible to see in Figure 2.6, the main threats are: malware (39.86% of studies), Data leakage of every kind (35.13% of studies), and Cyber attacks (16.89% of studies).

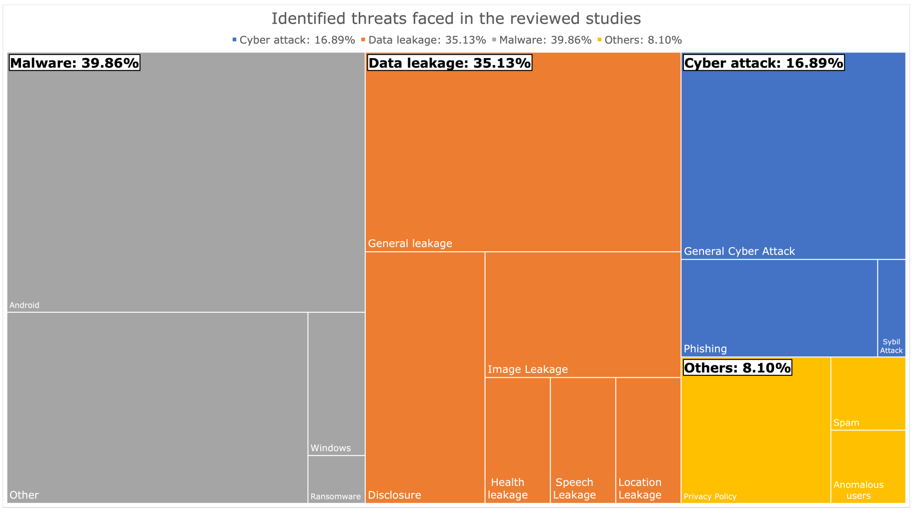


Figure 2.6: Main threats faced in reviewed studies.

Malware G. McGrow et al. [232] gave a defintion for malware which is *"any code added, changed, or removed from a software system in order to intentionally cause harm or subvert the intended function of the system"*, which informally can be seen as "any unwanted software". Malware intends to earn profit by stealing privacy or causing physical damage in the ICT system they're coded for [233]. For what concerns malware we found that Android is currently one of the main targets for malware threats, due to the increasing success of such OS platform on a large number of heterogeneous devices (the second place is Windows for the same reasons), ranging from smartphones to cars (android auto), smart TVs and wearable devices such as Google glasses. Such malware can perform malicious actions to compromise the device integrity, as well as to steal private and sensitive data, such as login credentials, contacts, text messages, camera images etc. and/or behave in a malicious way by subscribing expensive premium services associated to the specific activity of the involved devices.

We have also found a work specifically focused on detecting 0-day ransomware [98]. Ransomware is malicious software that threaten the user's privacy, infects his/her computer and displays messages demanding a fee to be paid in order for his/her system to work again [234]. This class of malware is a criminal moneymaking scheme that can be installed through deceptive links in an email message, instant message or Website. The leading causes of ransomware infections are spam and phishing emails, and malicious websites and web ads[2]. It has the ability to lock a computer screen or encrypt important, predetermined files with a password. Ransomware is considered a major and exponentially growing threat in 2016[3], based increasingly on anonymizing payment methods (e.g., Bitcoin) and anonymous networks (e.g., Tor). The most commonly experienced strains of ransomware by ransomware attack victims was CryptoLocker, followed by WannaCry and CryptoWall[4].

In 2018, the average cost of cyber insurance claims caused by ransomware amounted to 229,000 U.S. dollars[5]. Ransomware accounted for 15% of cyber insurance claims in North America in 2018, ranking only behind hacker attacks which accounted for 21

[2]https://www.statista.com/statistics/700965/leading-cause-of-ransomware-infection/
[3]https://www.statista.com/statistics/494947/ransomware-attacks-per-year-worldwide/
[4]https://www.statista.com/statistics/700944/global-msp-client-ransomware-attack-by-ransomware-families/
[5]https://www.statista.com/statistics/667597/cyber-insurance-claim-cost-north-america-by-cause-of-loss/

percent of claims[6].

The credits for the popularity of articles tackling malware are to the availability of public datasets for experiments. Indeed, differently from other privacy threat here we have a load of dataset such as those available from Andro-dumpsys[7], MalGenome[8] and Contagio Minidump[9], VirusShare site[10].

Cyber attack For what concerns cyber attacks we found that one of the most covered is phishing [195, 193, 196, 201, 228, 217], hence why we gave it a specific categorization in our taxonomy (Figure 2.7 and Appendix .1). Phishing is the fraudulent attempt to obtain sensitive information or data, such as usernames, passwords and credit card details, by disguising oneself as a trustworthy entity in an electronic communication. Typically carried out by email and messaging, phishing often directs users to enter personal information at a fake website which matches the look and feel of the legitimate site. Phishing attacks continue to be of persistent and critical concern to users, online businesses and financial institutions. In the last years we have witnessed an increasing number of phishing attacks, for example in 2019 the number has increased by 65%[11]. A phishing website lures users into divulging their sensitive information such as passwords, pin numbers, personal information, and credit card numbers, and uses such information for financial gains. According to current estimates, the annual financial losses due to phishing attacks surpasses $3 billion. Especially, for users, a phishing attack can mean a lot more than just financial losses as the loss of sensitive personal information has long term future ramifications as well. Other specific cyber attacks covered include sybil attacks [185] (which are indeed somehow similar to

[6]`https://www.statista.com/statistics/667484/share-of-cyber-insurance-claims-north-america-by-cause-of-loss/`

[7]`https://ocslab.hksecurity.net/andro-dumpsys`

[8]`http://www.malgenomeproject.org`

[9]`http://contagiominidump.blogspot.com`

[10]`https://virusshare.com`

[11]`https://www2.deloitte.com/content/dam/Deloitte/sg/Documents/risk/sea-risk-cyber-101-part10.pdf`

phishing), an impersonation attack [235] in which a malicious node masquerades as a set of nodes by claiming false identities, or generating new identities in the worst case [236]. The specific study found focuses on the sybil attack in OSN. Here the adversaries can launch sybil attacks through a large number of "fake" users, which are called sybils. Sybil attack takes advantage of its high proportion of sybils among users of the entire OSN to increase the impact of multiple malicious activities, such as spamming [237], phishing attacks, illegal access to personal privacy information [238], malware distribution, and so on.

Data leakage With regards to the data leakage in general, we can observe that there are studies specifically oriented towards health, location and image privacy, three hot issues in the privacy domain.

Concerning the health, with the introduction and evolution of e-healthcare [239, 240], and the ever-growing demand of data has exposed medical records to a wide audience. This obviously results in an uncontrolled data leakage threatening users privacy that must be managed, that is data must be anonymized. And this is the focus of the identified studies dealing with health leakage [89, 115].

For what concerns the location, this is ever-more available through mobile devices sharing it for different purposes, hence is one of the more attractive data to process. In fact, even when not very accurate, location can help adversaries to reconstruct the movement pattern of a user, his/her network of friends and so on. And this is the main issue faced in in [188], as example.

Lastly, due to the large scale smartphone adoption and the social media era, images are spreading at an unprecedented pace[12]. The issue is that users are often unaware of the risks "hidden" in the dissemination of such contents, which can include personal or sensitive data, private data of other individuals, and so on. Image privacy is a really challenging problem given, differently

[12] https://www.statista.com/chart/10913/number-of-photos-taken-worldwide/

from other types of leakage, data to protect is something that the users want to disseminate in the majority of the cases; moreover, OSN and media fundamentally encourage users to share everything to improve their presence in the virtual world. As a lot of private information are buried in images (and also messages), adversaries could navigate/crawl/search the disseminated contents to re-assemble scattered pieces of sensitive information. For this reasons, works like [181, 183] have focused on a image privacy protection method whose outcomes are invisible to human eyes, while [177, 91] have focused on methods to nudge users towards a more privacy-oriented behavior while disseminating images on the Internet.

Related to such a issue, we have also found several works dealing with the dissemination of personal or sensitive data mainly in text format on the Internet [174, 175, 178, 182, 21, 187] which represents one of the most covered issue.

Another interesting sub-threat covered is that of privacy policies, which are often neglected by users who only “accept” everything without reading and knowing what they agreed upon. This happens due lack of comprehension (people do not fully understand what is written in the policies) but also consent because several online services have specific clauses which bind the user by the time he/her switch on an application or access a Website [241]. Several works faced the issue proposing awareness mechanisms [225, 198, 218] to ease the understanding of privacy policies. As for the case of malware, the popularity of this kind of works has been fostered by the Usable Privacy Policy Project[13] which has released several datasets of annotated privacy policies for researchers all over the world.

Lastly, the identified studies facing Speech leakage have dealt with a very recent problem. On the one hand, we have research currently pushing for speech data to improve online speech-based service. On the other hand, we have a growing amount of sensors in public spaces (stations, airports, but also entire cities) which passively collect data. In the first case, if attackers know

[13] https://www.usableprivacy.org

a specific user's speech data in speech data publishing, they can learn the sensitive information of the user by analyzing such data. Speech content directly reflects the user's personal information. For instance, if the user's speech data shopping apps are leaked, the attackers can easily learn the user's job, preference, living habit [102]. Furthermore, by detecting the speaker's voice, the attackers can also obtain the privacy information of the user, such as gender, age, location religious faith, and so on. This is, indeed, the problem arising in the second case, were people, although spaces are public, may not fully realize the degree to which they may be recorded by the IoT devices, which is as a violation of expected privacy. Such records, if properly analyzed by an attacker can lead to speaker's gender or age detection [116].

For what concerns a more general point of view on data leakage we found [219] which focused on the infamous problem of the unfair transfer of personal data to third part services which track users' online behavior. Data leakage on the Web, indeed, is still a hot issue in the privacy field, just think that only in 2020 Safari Web browser, developed by Apple Inc., has been updated to filter out (with blacklist mechanisms) suspicious tracking resources while browsing[14].

2.4.3 RQ3: Which ML methods are the most utilized to protect users' privacy?

The objective of this section is to highlight the most used and effective ML methods exploited by the surveyed studies. We organized them in a taxonomy composed of five main classes, i.e., *Neural Network*, *Ensemble method*, *Support Vector machine*, *Clustering*, and *Other*. The latter includes ML methods less adopted respect the methods included in the previous classes. The taxonomy is organized as follows.

- *Neural Network*: it includes Autoencoder, CNN, GAN, LSTM, and MLP;

[14] https://support.apple.com/it-it/guide/safari/sfri40732/mac

- *Ensemble method*: it includes RF, DT, Fusion models, Adaptive Boosting, and XGBoosting;
- *Support Vector Machine*: it includes SVM and SVR;
- *Clustering*: it includes K-means, Optics, PCA, and Spectral;
- *Other* in which we have: NB, LR, , K-NN, Linear Regression, Reinforcement learning, Evolutionary methods, Ridge Regression, Latent Dirichlet Allocation, Word Embedding, Haar Cascade, Prediction by Partial Matching, Probabilistic-based classifier, and RIPPER.

The "Fusion models" set of ML methods, sub-class of the *Ensemble method* class, refers to all those works that have used multiple ML methods for the final prediction. This category includes: *(a)* features fusion, such as a ML method using transformed features of different kind output of multiple underlying ML methods, or coming from different data source and merged into a single-space by means of other ML methods; *(b)* ML methods based on voting where the inputs are the predictions made by other ML methods. These methods stem from the ensemble methods stacking, bagging and boosting, hence the inclusion into such class.

An overview of all these ML methods is presented in Table 2.8, while their adoption is shown through a treemap depicted in Fig. 2.7. It is worth to note that, some works have adopted as final solution a combination of k ML methods: in that case, we have considered each of k ML methods individually in the creation of the treemap. For example, if the work W has used *3* ML methods, MLP, RF and SVM as final solution, we incremented by one each of the three values, respectively.

We can observe that Ensemble methods represent the class with the most used ML methods (32.47% of the studies), followed by Neural Networks (27.27%) and SVM (16.02%). In addition, with regards to the first class, we found out that RF, is the most used ML method (37 studies) followed by DT (19 studies). This result is probably due to the easiness of these methods and the small set of parameters to tune for optimizing them. The same

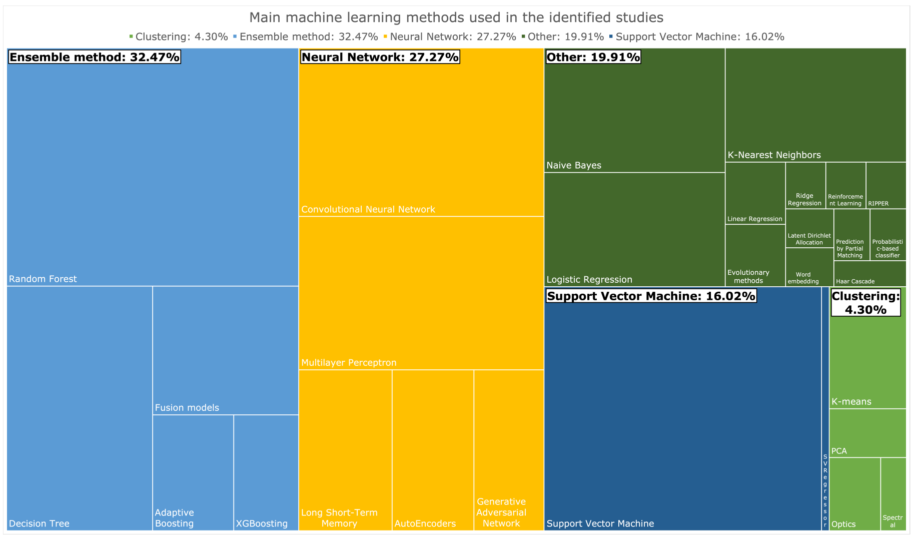


Figure 2.7: Main ML methods used in reviewed studies. Observe that several works employed multiple ML methods to provide final solutions.

explanation is valid for SVM-based methods, hence the third place among the most used ML methods as a whole. Indeed, with a deep inspection we found out that SVM is one of the most used single ML method among all the identified studies (36 studies). Among the most used Neural Network-based methods we find CNN (22 studies) and MLP (20 studies), vanilla or deep, followed by LSTM methods (8 studies). The interesting fact, is that one expects to find the use of CNN for problems involving images but actually we have found such use also with regard to other problems, e.g., malware detection [137, 123]. CNNs are so effective in classification that researchers have proposed techniques to translate not-images samples into images finding that CNNs outperform the other ML methods. Moreover, several ready-to-use CNN (as also deep MLP) can be found online[15].

In general, we observe that the studies employing Neural Network-based methods are extremely focused on such, while the

[15] https://keras.io/api/applications/resnet/

studies that used Ensemble-based methods have either compared several ML methods available in the literature finding that ensemble ones are better or focused more on highlighting the idea for facing the privacy threat and solve the problem than the specific ML method to implement (thus, using, for example, RF for its efficiency and fast tuning). This second case holds true for the studies adopting NB and LR methods. Lastly, unsupervised ML methods deserve to be mentioned, indeed K-means (5 studies) and AutoEncoders (7 studies) have been particularly popular among the identified studies for finding similarities between malware or network anomalies.

For an overview of the main ML method used by each identified study see Table 2.8.

2.4.3.1 SQ3.1: What kind of protection do such ML methods deliver?

The objective of this section is to highlight the different kind of privacy protections (see Section **??**) delivered via ML methods by the surveyed studies. Table 2.9 shows the type of protection delivered by each identified study.

In Figure 2.8 we can observe that the majority of identified studies deliver enforcement protections (roughly 59% of total) powered/enabled by the underlying ML methods seen in Section **??**. Enforcement protections come primarily from those works facing malware, which are built to illegally transfer wealth from users to the adversaries, and they are also employed against online tracking, a widespread unfair practice which should be blocked, and cyber attacks, which must be blocked once detected. Nudges are expressly delivered by (roughly) 28% of the studies, especially in form of awareness for users concerning for example anomalous users (alerting that a user is behaving in a suspicious way), or privacy policies ("explaining" what is written in the clauses, warning users against thid-part data processing, and so on). Another part of the identified studies (roughly 12% of total) did not explain how the ML method built could be used in real world scenarios.

ML method	**Studies**
AdaBoost	[145, 147, 159, 195, 231]
AutoEncoders	[230, 95, 140, 93, 123, 105, 85]
Clustering	[115, 101, 145, 86, 171] [101, 222] [209, 94] [101]
CNN	[176, 179, 130, 228, 184, 203, 204, 93, 206, 148, 208, 210, 212, 133, 214, 165, 137, 227, 100, 97, 169, 181]
DT	[125, 108, 146, 114, 155, 197, 163, 142, 109, 150, 145, 157, 126, 220, 221, 168, 136, 121, 102]
Fusion models	[158, 194, 129, 109, 195, 147, 186, 223, 161, 98]
GAN	[90, 181, 183, 104, 103, 96]
K-NN	[191, 131, 146, 114, 205, 154, 142, 196, 147, 138, 168]
LSTM	[228, 21, 188, 217, 122, 120, 189, 185]
MLP	[125, 192, 88, 93, 146, 219, 109, 132, 110, 114, 112, 141, 199, 116, 211, 94, 91, 207, 149, 151]
RF	[191, 87, 155, 125, 192, 131, 188, 162, 180, 114, 197, 139, 200, 184, 202, 94, 213, 153, 150, 99, 107, 142, 143, 219, 229, 147, 111, 117, 216, 156, 220, 221, 113, 182, 168, 121, 190]
SVM	[125, 192, 131, 177, 193, 124, 146, 180, 114, 197, 135, 92, 134, 94, 200, 175, 178, 215, 107, 147, 225, 145, 154, 144, 156, 162, 163, 216, 220, 221, 138, 166, 109, 168, 160, 173] [118]
XGBoosting	[196, 170, 156, 119]
Others	[127, 106, 201, 128, 172, 164] [167, 224] [174] [192, 118] [192, 124, 180, 89, 211, 170, 107, 218, 152, 225, 173] [198, 125, 131, 146, 187, 200, 150, 145, 107, 216, 220, 221] [226]

Table 2.8: Identified studies split by ML method employed.

Lastly, only two of the identified studies, i.e., [166, 184], delivers both nudge and enforcement solutions to protect users' privacy.

2.4.4 RQ4: Which research gaps and trends for future research are observable in the literature?

The objective of this section is to depict the main gaps in the research on the use of ML to protect users' privacy, in order to

Protection	Identified studies
Enforcement	[193, 89, 133, 138, 229, 137, 213, 228, 181, 183, 214, 116, 163, 165, 215, 211, 154, 230, 229, 169, 112, 231, 95, 85, 143, 185, 119, 123, 120, 121, 109, 118, 217, 101, 102, 103, 226, 104, 105, 106, 96, 167, 216, 98, 172, 158, 171, 160, 161, 162, 128, 142, 149, 168, 212, 192, 145, 125, 195, 191, 176, 127, 194, 129, 146, 132, 110, 114, 203, 202, 204, 206, 134, 148, 141, 94, 135, 199, 205, 153, 150, 200, 208, 156, 159, 92, 170]
Nudge	[124, 86, 174, 108, 111, 177, 175, 178, 196, 225, 136, 198, 201, 113, 99, 115, 164, 152, 117, 182, 21, 187, 189, 218, 97, 190, 222, 223, 144, 131, 180, 179, 130, 207, 147, 157, 220, 221, 186, 224, 100, 173]
Nudge & Enforcement	[184, 166]
Not Available	[126, 87, 209, 155, 91, 188, 88, 197, 90, 139, 209, 107, 140, 210, 151, 227, 122]

Table 2.9: Identified studies split by typology of protection delivered.

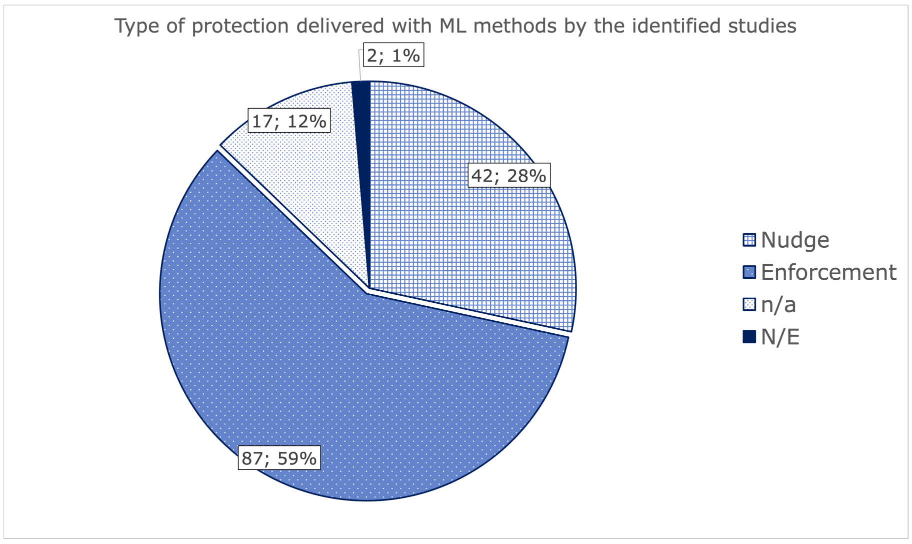


Figure 2.8: Type of solution delivered with ML methods by the identified studies. n/a = not available (when not clear from the paper), N/E = both Nudge and Enforcement are delivered.

provide insights and new directions for future works.

Figure 2.9 quantifies the number of reviewed works based on the type of outcome, that is "Study", "Framework", or "Tool". We see that Study represents the majority class with roughly 69% of the total, followed by Framework with roughly 18%, and lastly Tool with roughly 13%.

Type of work	Identified studies
Study	[128, 86, 195, 87, 108, 111, 177, 197, 175, 196, 199, 202, 94, 213, 198, 201, 92, 211, 209, 138, 133, 89, 225, 229, 137, 181, 163, 170, 165, 21, 91, 214, 107, 99, 93, 117, 154, 230, 95, 228, 231, 116, 113, 164, 169, 182, 183, 85, 185, 187, 188, 119, 121, 151, 120, 122, 123, 226, 217, 186, 104, 105, 106, 96, 98, 172, 190, 158, 160] [191, 192, 194, 179, 127, 114, 132, 206, 180, 203, 134, 130, 204, 110, 205, 141, 140, 150, 144, 159, 222, 166, 227, 100, 212, 210, 215, 145, 157, 216, 167, 223, 220]
Tool	[174, 136, 178, 184, 155, 115, 152, 118, 218, 219, 162, 189] [125, 135, 149, 143, 156, 221, 168, 97]
Framework	[124, 126, 193, 112, 109, 148, 101, 102, 103, 161] [176, 88, 131, 129, 146, 200, 90, 139, 142, 208, 153, 207, 147, 224, 171, 173]

Table 2.10: Identified studies split by typology of work.

In the light of the above, we notice that the Technology Readiness Level (TRL) [242] of all the reviewed works is quite low, mainly in the range 1-4 (from the observation of basic principles to a validation into laboratory environment). Indeed, an interesting aspect to observe is that among all the studies reviewed only a few of them are real tools and even less have released a public repository or disseminated in other ways the software described. For what concerns tools we only find those highlighted in Table 2.10, and the disseminated software (available on GitHub or GitLab for example) are only those described in [178, 104, 179, 220, 145, 140, 222, 130, 142, 151]. It emerges that, from this point of view, the research in use of ML to protect users' privacy has yet to make many steps forward, compared to what happens, for example, in the world of ML applied to sports [84] where the TRL levels of the proposals is higher. This result supports also what we have found concerning the type of solutions delivered by the identified studies (see Figure 2.8). Given a big chunk of the studies are placed in a low TRL level, they can be considered work-in-progress to be refined in future steps. Therefore, it is clear that some of the authors did not faced the problem of delivering a "final" solutions to users, that is why for 12% of them there was not a clear statement in the paper.

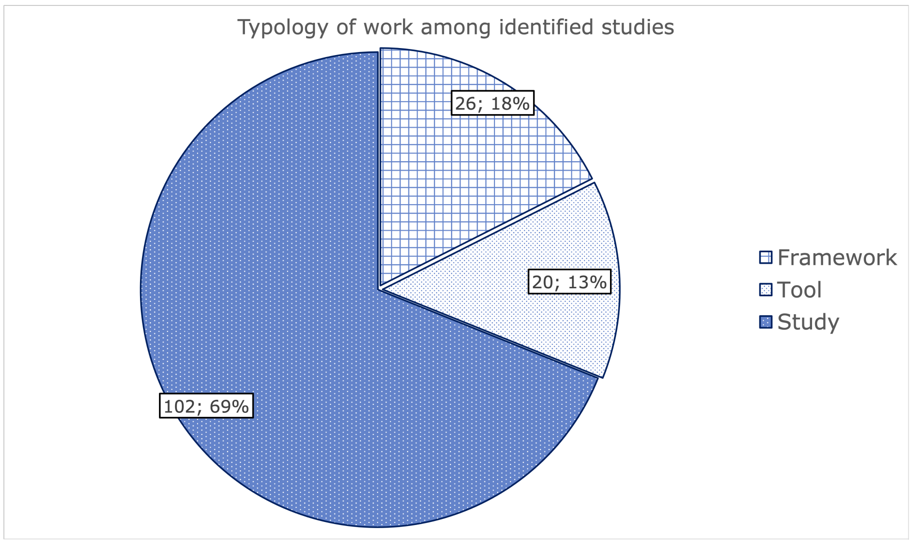


Figure 2.9: Typology of work among identified studies split into Framework, Tool, and Study.

Therefore, we encourage researchers to also validate their methods and results in the real-world. If the field is to gain widespread validity such research is needed, and researchers should try to get in contact with final users more. Related to this issue, we have found: *(a) lack of users' study*: only a few works performed a users' study such as a usability test: it is not clear, indeed, whether users will ever adopt the results and the protections conceived, whether they like it or not; *(b) lack of comprehensive comparison*: a small percentage of the works showed a robust comparison section where the proposed results were compared against those found in the literature or the commercial solutions[16].

Furthermore, among the identified studies we cannot find specific solutions and protections designed for certain categories of

[16]In this respect, we have only found [219] comparing the proposed tool's results with commercial solutions, and [167] comparing malware detection systems against Assemblyline, an open source tool.

users. For example, we can make the case of children. The GDPR[17] limits the contents that can be shown to 13 to 15-year-old users and in art. 8 states the solutions to protect children's privacy should be tailored since the consent for data processing must be expressed by the the holder of parental responsibility. On the same page, for example, the protections relying on nudge should tailor the nudges for children. The same holds true for other privacy threats and other users categories: another example is the vast literature on IoT for elderly, which, among the identified studies, has not a counterpart concerning the privacy protections for elderly in the IoT. Last but not least, we have not found privacy solutions designed for multiple recipients and when they protect one, i.e., the user, perhaps they warn the other, i.e., a public or private body. Their outputs can inform policy and decision makers, support automated, human or hybrid fact-check efforts, improve privacy, and guide the definition of legal frameworks.

Lastly, we invite researchers to disseminate the data crawled and/or used in their works. If it was not for the studies facing malware employing public repositories, and for the studies dealing with PP (see Section 2.6), only a negligible percentage of the identified studies has disseminated the datasets employed. And this is a clear limit to the development of the research in the use of ML to protect users privacy. We are aware there are some datasets that are difficult to publish precisely for privacy reasons and these could also be made available with the conscious use of anonymization techniques, but there are data that could be made available easily which are not. This represents a longstanding issue, as old as the research on Big Data in general, seeming to taint not only the research in privacy (as also emphasized in previous survey [43]), but also other fields, e.g., sports. In fact, a 2020 study on the use of data anlytics in sport highlighted the same problem [84]. This question is linked with *(b) lack of comprehensive comparison*, because when data are not available to researchers, the comparison between published studies becomes prohibitive (repeat all the procedure conceived in the other studies) if not impossible given

[17] `https://gdpr-info.eu`

the amount of data with which usually ML methods are fit.

2.5 Limitations

In this article, we tried to mitigate the research bias and sought to answer the research questions (Table 2.2) in order to obtain an outline of the current literature related to the use of ML to protect users' privacy without assessing any article that refers to other kind of effective privacy safeguards (those not based on ML methods). Therefore, this research is limited to aspects related only to articles that make use of ML to protect users' privacy rather than including/proposing new methods for malware or phishing detection, for example. The research was limited to obtaining articles published in scientific portals presented in Table 2.3 that are related to the use of ML to protect users' privacy. The systematic review used a rigorous methodology and it is limited to studies found on these websites, therefore it may miss some relevant article from other portals. We carried out the data extraction and discussed any disagreements before discard a work. In order to be extremely cautious in data extraction, the impurity, title, and abstract filters were made manually in our study. However, as appointed in other works [55, 56], manual searches may miss relevant articles.

2.6 Conclusion

The current work proposed a systematic literature review to frame the use of machine learning to protect users' privacy besides tracing back existing solutions to two fundamental approaches, that is *enforcement* (i.e., solutions which impose constrains and hamper breaches of norms) and *nudge* solutions (i.e., solutions which inform users and increase their awareness to promote privacy-oriented behaviors). For this purpose, we performed a systematic analysis of the relevant articles from January 2017 to October 2020. We then propose a taxonomy of such works that contains the main areas of application, threats faced, ML methods adopted, type of solution delivered. The taxonomy and review result made it possible to answer general and specify questions about the current status, challenges, open issues about the use of machine learning to protect users' privacy. Once we analyzed articles selected over the review, we noticed a broad set of findings and were able to identify some gaps, trending, and challenges regarding the protection of privacy. One of the future challenge is that of developing specific protections for specific kind of persons, and design solutions for multiple recipients: for instance, for the user (protecting him/her) and for third part bodies (warn policy makers about unfair practices). The analysis has shown that a few works proposed tools or frameworks for protecting users and the majority (at a low Technology Readiness Level) focused on the development of a novel ML method. We also noticed many studies did not released the datasets employed. The final review analysis presents a set of findings and research items that can be used as a direction for future research by the scientific community. To better elucidate them, we highlight:

- An overview of how to protect privacy;
- A new reading key for the research in the field of ML to protect users' privacy tracing back existing approaches and solutions to the categories of *enforcement* and *nudge*;
- Offering a big picture about the current use of ML to protect users' privacy;
- A taxonomy to classify the areas, threats, ML methods, type of protection delivered.

In future studies, we envision a focus on the challenges and issues related to the study, especially on developing proposed solutions so to scale up the TRL levels and examine how final users perceive and use the delivered solutions.

Chapter 3

Research design

The results obtained in the systematic literature review (Chapter 2) have highlighted the main threats (Section 2.4.2), gaps/limitations, and challenges (Section 2.4.4) for the research on the use of ML to protect users' privacy.

In the light of the above, this dissertation aims at expanding the research horizon by filling some of those gaps, declining ML in specific contexts and areas introducing novelties against the literature. Ideally, the research can be split into two parts. The first one (Chapters 4; 5) concerns issues strictly related to the privacy protection such as Web tracking and the unaware/uncontrolled disclosure of personal and private data online. The second part (Chapters 6; 7) concerns issues and problems "neighbouring" (often intertwined) to the privacy, deserving adequate attention and exploration because the privacy protection is clearly part of a broader issue, which is that of rights protection in the digital society. The issue involves rights even more crucial than privacy (e.g., children rights, fundamental rights like right to non-discrimination) which are threatened through the violation of the privacy, in broader sense understood. Indeed, by violating a citizen's privacy it is possible to gather data which once used to fed sophisticated analytics software can turn (willingly or not) into the trigger to intrude his/her fundamental rights, for example forming a biased outcome that systematically discriminates people based on their race, ethnicity, religion, political preferences, gender or sexual orientation. It is no accident that GDPR goes beyond the Charter of Fundamental Rights of the European Union[1] (CFREU), explicitly aiming to protect all the fundamental rights and freedoms that are implicated by the processing of personal data. Therefore, this dissertation overlooks on threats to privacy and related rights that can be faced properly adopting ML.

In the following, we will present a series of research activities aiming to support *privacy*, *child* and *consumer* protection in the digital society. Each activity complies with the following schema:

1. *Frame the research scenario*: we start depicting the specific scenario and challenges to face.

2. *Requirements elicitation*: an open dialog with domain experts taking into account technical and legal aspects allows to define the solution design. We also include insights from (strictly) related works not falling into the systematic literature review due to the rigorous methodology applied.

3. *Experiments*: we explore the use of ML techniques to pursue the research objectives compliant with the aforementioned requirements. In more details this phase includes the following main points: *(a)* dataset collection, *(b)* features extraction and data labeling, *(c)* applying ML techniques to provide a proof-of-concept or a validation with a relevant dataset, *(d)* evaluation of obtained results. When possible, we iterated the phase.

4. *Artifact development (and evaluation)*: we put the ML solutions designed into practice developing (prototype) tools for final users. Every tool is evaluated in real-world in terms of efficiency and effectiveness aiming to answer the questions: *Q1: "how does the tool impact on the users experience?"* Such as, it slows down the device used; *Q2: "does the tool provide the correct output?"* Such as, it correctly classify a

[1]`https://www.europarl.europa.eu/charter/pdf/text_en.pdf`

Web tracker. When possible, we perform studies aimed at evaluating the tool from the users point of view, including also the intuitiveness and pleasantness of User Interface, and the likelihood to use it in the future.

5. *Dissemination*: the research major results have been submitted to top quality journals according to Scimago[2] and ANVUR[3]. Minor results and proof-of-concepts have been submitted to appropriate conferences.

In order to increase the intelligibility of the different research activities, we will provide the *Chapter Highlights*, with the key points (3 to 5) of the presented works, at the beginning of every chapter, as done already for Chapter 2. We also provide the (tentative) classification of the solutions proposed in the chapters according to the taxonomy depicted in Chapter 2, that is based on *Type of work*, *Threat*, *ML Method*, *Type of protection*.

[2] `https://www.scimagojr.com`
[3] `https://www.anvur.it`

Chapter 4

ML and privacy in practice: enforcement

Chapter Highlights

- Proposal of a proof-of-concept of the use of ML to classify between malicious and functional resources on the Web based on HTTP features;
- Definition of a ML approach to classify between malicious and functional resources on the Web based on HTTP features and manual engineered features of JavaScript programs validated on a large dataset;
- Introduction of a novel *hybrid* mechanism for filtering third-party resources based on ML and blacklisting which adopts ML to update the blacklist;
- Development of such a mechanism in *GuardOne*, a Google Chrome extension *enforcing* privacy by blocking unwanted content on the Web;
- Real-world evaluation of *GuardOne* in terms of effectiveness and efficiency in comparison with similar commercial privacy tools showing that *GuardOne* is at least on par with commercial solutions offering at the same time better performance (less RAM and CPU usage).
- Classification according to the taxonomy depicted in Chapter 2: **{ Type of work**: Tool, **Threat**: GenLk, **ML Method**: RF and MLP, **Type of protection**: Enforcement **}**

4.1 Introduction

Today, it is a common practice for websites to rely on services provided by third party (3rd-party, from now on) companies to monitor daily activities of Internet users, with the main goal of providing online personalized experiences. To this aim, almost everything about our lives, in terms of any interaction with technologies, is recorded and stored to create data: Google searches, credit card purchases, Facebook post, social network activities, medical information, or Netflix preferences. In addition, voter registration, medical history, cellphone geolocalization, data from mobile App usage, represent useful data to amplify the set of information collected and maintained for each individual user. This data is analyzed and used by the entities after collecting it. Specifically, 3rd-party companies, that include aggregators, analytics companies, marketing companies, or commercial data brokers, use different vectors and technologies to monitor Web users such as: advertisements, metrics and analytics functionalities, social widgets, Web bug, embedded JavaScript libraries (often hosted on CDN owned by first party sites), standard HTTP headers and so on [243].

The practice of monitoring daily activities, well-known as online behavioral advertising, although increases the effectiveness of the marketers' campaigns as well as their revenues ($28.4 billion in the first quarter of 2019) [244], it has also strong implications for both users and performance.

The first implication is about the *privacy* of individuals, at risk while browsing the Web [245] and, in general, while interacting with technology [246, 247, 248]: indeed, behavioral advertising heavily relies on the use of valuable information that could lead to an accurate reconstruction of users' interests profiles, when leaked to 3rd-party entities [249, 250, 251]. Therefore, the main problem is not strictly related to the information collection, but to the uncertainty associated with how personal information (e.g., full name, date of birth, address and zip code, email address and so on) and sensitive information (e.g., political and religious views, health data, biometric and genetic data and so on) is handled by online establishments and who access access to it. Moreover, the proliferation of these publicly available information online, combined with increasingly powerful computer hardware and with efficient data mining techniques, has grave privacy and policy implications since it could made it possible to re-identify anonymized data [252, 253]. As result, pseudo-anonymous data, linked with personally identifiable information, could be potentially used for secondary activities, such as identity theft, social engineering attacks, online and physical stalking and so on [254, 255, 256], or simply sold for marketing/political campaigns (as happened with the Facebook/Cambridge Analytica scandal[1]). Users have expressed their concerns regarding this practice [257] and several tools have been provided to control behavioral advertising and, indirectly, the information leakage, and limiting their effects. Client-side tools, in the form of browser extensions became popular mainly because the difficulties of setting and preserving opt-out cookies [258] and the fact that the Do-Not-Track HTTP header mostly is ignored by Websites [259]. Some of the most popular client-side tools exhibit high variation in the effectiveness when preventing requests to 3rd-party domains [260], others are not able to fully block fingerprinting services [261] and suffer of lack of effective protection on mobile devices. In addition, in [262] authors showed significant usability problems of some popular privacy tools. However, it is also well-known that many popular ad-blockers (including AdBlock and AdblockPlus) participate in the Acceptable Ads program, allowing *non-intrusive* ads to be allowed [263].

The second implication is about *performance* and specifically, the more computation/communication (as well as more energy when considering mobile devices [264, 265]) required in order for downloading Web pages and associated behavioral advertising. Blocking trackers with browser extensions comes at the cost of additional memory and CPU overhead for matching requests against their blacklists [261, 265]. Most of the tools rely on manually

[1]https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html

maintained blacklists, which need to be continuously updated to protect users in an effective way. Trackers rise at a rapid pace implying a quick evolution of the advertising and analytics landscape. As a consequence, newly trackers have to be added to the existing blacklists, or entries in these files have to be updated with the new rules. These rules are often represented as regular expressions (coded in standard languages or in proprietary formats), where each expression corresponds to the new tracker or object to block[2].

The challenges we addressed in this chapter concern on the one hand, the size of the blacklists whose management represents a problem for performance (mainly in terms of system memory usage) and, on the other hand, the arduousness (human intervention) of keeping the blacklists up to date, in order to make them effective against the new aggregators entering the advertising landscape. The idea of relying on an automatic approach to classify 3rd-party resources with the final goal of avoiding the use of blacklists manually built and maintained is not new [266, 267, 268, 269]. However, although these works present different approaches for resource filtering, none of them provide a real countermeasure that deploy it, providing evidence of its effectiveness during "*online*" activities. An overview of the proposed system is available in Figure 4.1

Specifically, the main contributions of our work can be summarized as follows:

- To propose a ML approach to classify between trackers and functional resources on the Web based on HTTP traffic features and show a proof-of-concept of the envisioned solution on a dataset of manually labeled samples.

- To define a ML approach to automatically classify unwanted content; this approach can be used for defining classifiers able to distinguish between malicious (or tracking) requests

[2]Here you can find the EasyList, the most popular filtering list maintained by AdBlock Plus, https://easylist.to/easylist/easylist.txt

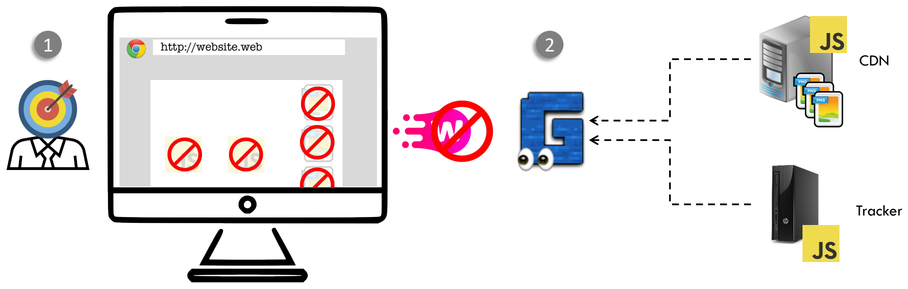


Figure 4.1: An overview of the proposed system. (1) The user browses a Website that requests for several malicious resources (trackers, advertisements, and so on). (2) Our system, *GuardOne*, analyzes the resources and blocks them enforcing user's privacy.

and functional (or needed for a proper functioning of a website) requests. The approach is validated on a large dataset of automatically labeled samples.

- To propose a *novel hybrid* mechanism for filtering 3rd-party requests from users' browsing sessions; the mechanism is hybrid in the sense that it exploits both the blacklisting technique (widely used in this field) and a ML classifier defined according to the approach we proposed in this chapter, to automatically classify contents, fill in and/or to update the blacklist with the tracking requests, by filtering out them from the user browsing session.

- To provide an implementation of this hybrid mechanism, named *GuardOne*, as a prototype of a Google Chrome browser extension; the system is *customizable* in the sense it makes use of *tiny* and *customized* blacklist built according to the users' browsing habits and, therefore, according to their real dangerousness; the size of these blacklists is smaller than those used by the most popular privacy tools, since they take into account the habits of a single user; we developed a client-side solution in JavaScript programming language implementing a ML-based approach and exploiting tiny and not pre-built blacklists but populated according to users behaviors; we chose Google Chrome given it high popularity[3]; we remark that, to the best of our knowledge, *GuardOne* is a first prototype of a Google Chrome extension for protecting privacy on the Web that relies on ML techniques.

- To provide an exhaustive evaluation of *GuardOne*, in terms of performance and effectiveness of its filtering, comparing it with the most popular tools in the same field. To this aim we implemented a measurement framework to automatize all tasks, from the construction of the workloads to the analysis of the final results. To the best of our knowledge this is the first example of ML-based privacy approach implemented as software prototype, and tested "*In vivo*" [265] (with users' real browsing sessions) to analyze both effectiveness and impact on the user's experience.

This chapter is organized as follows. In Section 4.2, we introduce the background needed to understand the rest of the chapter and we describe the most popular commercial privacy tools. In Section 4.3, we propose a proof-of-concept on the use of ML to distinguish between trackers and functional resources on the Web. In Section 4.4, we present a development of the research by introducing a ML approach for filtering 3rd-party resources validated on a more relevant dataset. Then, we provide details about the prototype tool we implemented, named *GuardOne* (Section 4.4.2) and, in Section 4.4.3, we present the methodology and the results of an experimental study aiming at comparing *GuardOne* with its popular commercial competitors in terms of performance and effectiveness. In Section 4.5 we discuss some interesting work in this field facing similar specific challenges. Finally, in Section 6.6 we conclude with final remarks and future directions.

[3]The data from StatCounter shows that Google Chrome is the leading browser, with about 63% of marketshare worldwide.

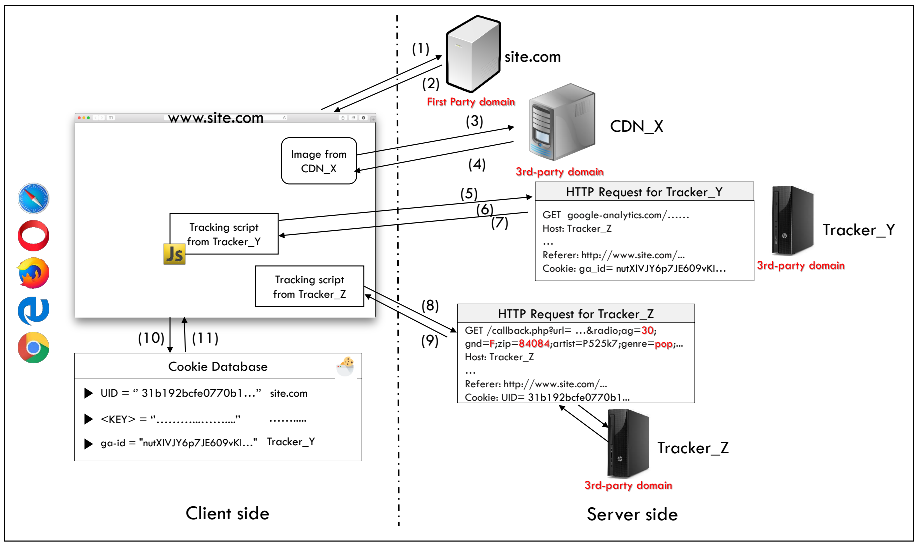
4.2 Preliminaries

In this section, we provide information needed to understand the rest of the chapter and we describe the most popular privacy tools available in literature.

4.2.1 JavaScript programs and HTTP requests

Web pages consist of several resources embedded which have to be fetched before a Web page can be displayed. Some resources are retrieved from the visited website, while many other are retrieved from 3rd-party sites.

In Figure 4.2 we show the typical Web tracking scenarios performed by JavaScript programs and how personal data could silently transmitted through headers of the issued HTTP requests.



```
GET  google-analytics.com/……
Host: Tracker_Z
…
Referer: http://www.site.com/...
Cookie: ga_id= nutXIVJY6p7JE609vKI…
```

```
GET /callback.php?url= …&radio;ag=30;
gnd=F;zip=84084;artist=P525k7;genre=pop;...
Host: Tracker_Z
…
Referer: http://www.site.com/...
Cookie: UID= 31b192bcfe0770b1...
```

- UID = " 31b192bcfe0770b1…"   site.com
- <KEY> = "…………..……….…"   ……….....
- ga-id = "nutXIVJY6p7JE609vKI…"   Tracker_Y

Figure 4.2: Examples of tracking behaviors through the execution of JavaScript programs and the leakage of personal information silently embedded into HTTP headers.

The user types the URL of the requested Web page; upon fetching the Web page from the first party domain (steps 1 & 2), the Web browser interprets all the HTML tags within the page, including the ones referring to JavaScript programs. The JavaScript programs execution enables the browser to send requests to different sites from 3rd-party domains. The retrieved content could be considered as *functional* for the Web page, e.g., fetching content from a CDN (steps 3 & 4) or as *tracking*, i.e., advertisements, analytics and so on. With regard to tracking behaviors, a JavaScript program, an analytic service as an example, embedded in the website, after loading (steps 5 & 6), can set a 3rd-party cookie and then communicate back to the 3rd-party server with other tracking information (step 7), and obtain a profile of the user activity on the website [270]. Additionally, a JavaScript program, after loading, can send user's personal information to 3rd-party services [271] (steps 8 & 9). It can add users information in the *Referer* header (such as personal ID), in the *Request URI* or through *cookies* [272]; they can also use Web bugs [273] [274] or send data through images, HTML `iframe`, `embed`, `object`, and other tags. Finally, 3rd-party programs can read from or write to cookie database [275] [276] to monitor users (steps 10 & 11).

We want to remark that we define *"malicious"* any tracking resource that aims at monitor users' activities, while we define *"functional"* any resource needed for the proper functioning of a web page. Moreover, we have also to highlight that very often functional resources can be served by Content Distribution Networks (CDNs). A CDN is a geographically distributed network of proxy servers and its data centers. Their goal is to provide high availability and high performance by distributing resources and services as close as possible to end users. They serve a large portion of the Internet content today, including web objects (graphics, images, and scripts), applications, live and on-demand streaming media, and so on. They may provide needed content for the correct functioning of websites even if they represent 3rd-party domains, and therefore, they can be erroneously classified as trackers.

4.2.2 Privacy Protection Tools

Here we briefly describe the privacy protection tools (PPTools) we used in the comparison study described in Section 4.4.3. All PPTools, except for Privacy Badger, are exclusively based on a blacklisting-based approach and are available for both desktop computers and mobile devices.

AdBlock Plus [277] is the most used browser extension to remove unwanted content from the Internet, including annoying advertisements, bothersome banners and troublesome tracking. By default, it relies on the *EasyList* ruleset (https://easylist.to/), that consists of almost 70,000 entries, with a large number of community-maintained rules. Rules are written in a custom syntax and translated to regular expressions [278]. Other subscriptions lists are available for regional blocking or to add other types of resources, such as web bugs, tracking scripts and information collectors (*EasyPrivacy* list). The acceptable program is enabled by default for the AdBlock Plus browser extension, and we disabled it for our experiments.

Ghostery [279] is a proprietary software that provides control over advertisement and tracking technologies. Launched in 2009, Ghostery has more than seven million monthly active users and a centralized database with more than 2,000 entries, (i.e., trackers). By default, Ghostery only provides feedback on which trackers are included in first-party websites, without blocking them, therefore, for our comparison, we activated the blocking for all 3rd-party available categories (Advertisements, Analytics Services, Social Media Widgets, and so on).

Disconnect [280] is, in its basic version, a privacy tool that blocks 3rd-party tracking cookies and JavaScript. Pro and Premium versions are available to block trackers and malware as well as to protect VPN networks. Similar to Ghostery, it uses a centralized approach to create blocking rules, with the database maintained by the company behind it.

No-Script Lite [281] blocks JavaScript programs (inline and external), Java programs, Silverlight, Flash and other executable content on a web page that may undermine security, including tracking. The default setting provides blocking script only with CSP (Content Security Policy) method. Similar to AdBlock Plus, rules are expressed as regular expressions.

Privacy Badger [282] blocks 3rd-party tracking (identifying cookies, local storage supercookies, and canvas fingerprinting) leveraging on own heuristics (mostly based on cookie entropy) as well as on internal blacklists. The main difference with respect to other PPTools is that it does not require any settings, knowledge, or configuration by the user. To block advertising, it sends the Do Not Track header with each request (even if it is well-known that it has a little impact on filtering [283], or that websites often simply ignore it), and evaluates the likelihood that the user is still being tracked. If the algorithm deems the likelihood is too high, it automatically blocks the request for the 3rd-party domain.

We trained Privacy Badger with the Alexa top 100 websites to evaluate its effectiveness in terms of resource filtering.

Table 4.1: Client-side privacy tools: filtering mechanisms and statistics.

Browser Extension	**Filter-Rules**	**Entries Number**	**Active users**
AdBlock Plus	EasyList ruleset	Almost 70,000	10.000.000+
Ghostery	Proprietary	Over 2,000	2.835.632
Disconnect	Custom (GPL)	Over 2,000	897.515
No-Script Lite	Proprietary	–	52.281
Privacy Badger	Algorithmic	–	792.664

4.3 On Analyzing Third-Party Tracking via ML: a proof-of-concept

In this section, we propose the use of ML methods to classify trackers on the Web. Specifically, we focus on the feasibility of classifying every type of 3rd-party tracking resource. To this end, we first dowloaded a set of Web resources by browsing Alexa's Top 10 websites[4] for each category (shopping, sports, etc.), second we enginnered a set of metrics suitable to distinguish between functional and tracking resources; those metrics are based on HTTP traffic. In order to evaluate the effectiveness of the proposed metrics, as proposed in [284, 285, 286], we have compared the performance of several ML models based on supervised learning among the most used in literature, i.e., NB [287], RF [70], MLP [63], C4.5 [288], and SVM [65]. As best result, the Random Forest can classify functional and tracking resources with an accuracy of 91%.

4.3.1 Our approach for classifying Web resources: a proof-of-concept

In this section, we describe a novel ML approach to automatically detect 3rd-party tracking activities while browsing the Web. We remark that such a approach can be used to overcome the problems of having to manually update blacklist. Our classification problem is a binary classification problem on Web resources where positive samples are *tracking* requests and negative samples are *functional* ones. In the following, we describe in details the three main steps of our approach (see Figure 4.3):

- *Data collection*: by means of automatic browsing sessions we navigate a number of Web pages from Alexa's Top 10, meanwhile we trace the HTTP traffic.
- *Features extraction and labeling*: *(1)* a set of suitable features has been defined and engineered from the data collected, in order to build the dataset used for the training and testing of the ML models, *(2)* the dataset has been manually labeled.
- *Validation and testing*: the dataset has been split into a train set and a test set; a k-fold cross-validation has been

[4] `https://www.alexa.com/topsites`

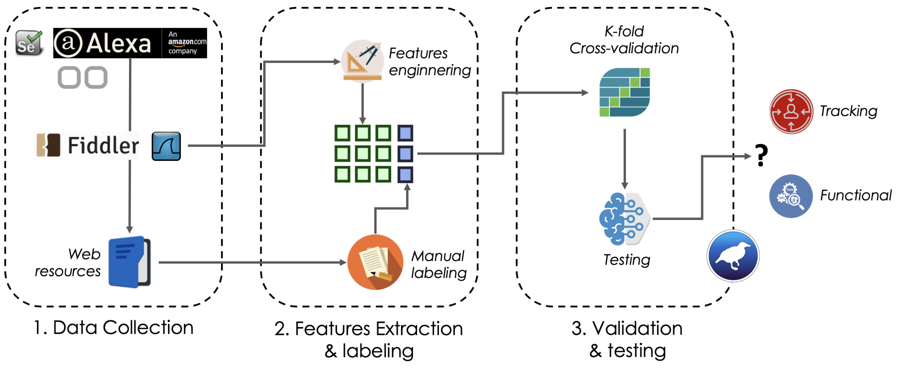


Figure 4.3: Steps of the approach for classifying Web resources.

performed on the train set to validate several well-known ML models previously described; once validated, machine learnign models have been tested on the test set.

Data Collection

By using Selenium webdriver [289] we automatically browse Alexa's Top 10 Web pages for each of the Alexa's categories[5], i.e., *adult, arts, business, computers, games, reference, regional, science, shopping, society, health, home, kids and teens, news, recreation, sports and world*. We then employ Fiddler [290] and Wireshark [291] to monitor and capture HTTP traffic while browsing such websites. Moreover, each captured resource has been downloaded. Specifically, in this phase we collected 1000 Web resources.

Features extraction and labeling

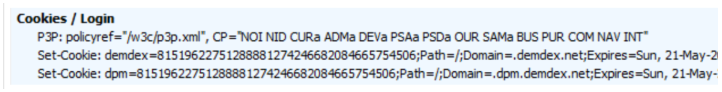
In this section, we provide details about the features engineered from the data collected (Section 4.3.1) and we explain how samples have been labeled.

Features. The choice of the set of features is one of the most significant step during the definition of a ML system because fea-

[5] https://www.alexa.com/topsites/category

tures must describe, in the best possible way, the input samples. In our case, given the kind of resource, the set of features has to describe the information about the HTTP traffic associated with. Features engineered are:

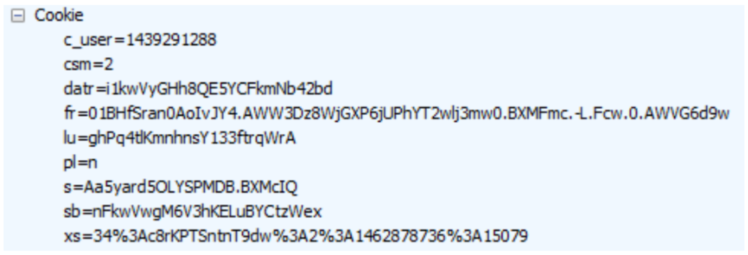
1. *Number of Set-Cookie* included in the HTTP request associated to a single resource: a huge number of Set-Cookie is used to send different information to more tracking domains (see Figure 4.4).



```
Cookies / Login
    P3P: policyref="/w3c/p3p.xml", CP="NOI NID CURa ADMa DEVa PSAa PSDa OUR SAMa BUS PUR COM NAV INT"
    Set-Cookie: demdex=81519622751288881274246682084665754506;Path=/;Domain=.demdex.net;Expires=Sun, 21-May-2
    Set-Cookie: dpm=81519622751288881274246682084665754506;Path=/;Domain=.dpm.demdex.net;Expires=Sun, 21-May-
```

Figure 4.4: An example of tracking request which set different cookies from demdex domain, specifically from both *dpm.demdex.net* and *demdex.net*.

2. *Number of cookies* sent by a resource at the moment of the request: tracking, analytics or advertisement resources usually send more cookies to send different user's information (see Figure 4.5 for an example).



```
⊟ Cookie
      c_user=1439291288
      csm=2
      datr=i1kwVyGHh8QE5YCFkmNb42bd
      fr=01BHfSran0AoIvJY4.AWW3Dz8WjGXP6jUPhYT2wlj3mw0.BXMFmc.-L.Fcw.0.AWVG6d9w
      lu=ghPq4tlKmnhnsY133ftrqWrA
      pl=n
      s=Aa5yard5OLYSPMDB.BXMcIQ
      sb=nFkwVwgM6V3hKELuBYCtzWex
      xs=34%3Ac8rKPTSntnT9dw%3A2%3A1462878736%3A15079
```

Figure 4.5: An example of several cookies set at request time by a tracker. A unique user id has been set (*c_user*) togheter with other information associated with him, like *datr* used to understand if the user access by multiple devices.

3. *Length of sent cookies*: the 80% of tracking cookies contains more than 35 characters [292], unlike the ones provided by

functional resources. This behaviour is justified by the quantity of information trasmitted to tracking domains.

4. *Cookie duration*: in order to learn more and more user's preferences, a tracker has to observe users habits for a long time. In [292] authors showed that the 90% of cases tracking cookies lasted more than 6 months.

5. *URL length*: information can also be sent through HTTP GET method, instead of cookies. In this case, data are sent as parameters in the URL, increasing consequently its length.

6. *Number of parameters in HTTP GET request*: this feature is related to the URL lenght, indeed tracking, analytics or advertisement resources tend to send more data, thus parameters to share different user's information.

7. *Sent/received bytes ratio*: in HTML pages, trackers insert some invisible resources, i.e., images or iFrames with a dimension of 0x0, 0x1, 768x0, etc. named Web bugs [293]. Such content does not have a functional aim for the webpage, it is actually used just to track the user's habit on the page. The Web bugs still send an HTTP request from the client to the server. Then the server takes the information about the user, embedded in the HTTP (cookie, referer, user-agent, and so on), and reply to the client with an empty frame or image. This result in a difference between sent bytes (information about the user) and the received bytes.

Labels. Samples have been manually labeled. Specifically, as proposed in [294], we asked to three experts in the privacy field to independently label the resources as functional (with label `0`) or tracking (with label `1`). We assigned the label according to the majority voting. Samples are evenly balanced between tracking and functional ones i.e., 500 has label functional while the latter 500 has tracking.

Validation and testing

In this section, we detail the validation and testing phase conducted by means of Weka [295].

We split (with a stratified approach) the dataset into: (1) the *train set*, obtained including the 80% of the samples (randomly chosen), and (2) the *test set*, obtained including the remaining 20% of the samples. Thus, the train set contains 800 samples (400 functional and 400 tracking), while the test set contains 200 samples (100 functional and 100 tracking).

We used a scaler to normalize the data according to the min-max technique,i.e.,

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

where x_i is the true feature value for i-th sample, z_i is the normalized value, and $min()$ and $max()$ compute minimum and maximum value respectively for the feature x.

Then, we first validate the RF, J48, NB, MLP, SVM models (see Chapter **??**) in a `k`-fold cross-validation phase on the train set, second we perform the testing phase on the test set (i.e., unseen samples). `k`-fold cross-validation is a resampling procedure employed to evaluate ML models on a limited samples. The function has a single parameter called `k` that specifies the number of groups that a given dataset (train set in our case) has to be split into. We set $k = 5$ and $k = 10$. Informally, the function takes the first of the `k` groups as validation set and the remaining `k-1` as train set [296].

Experiments have been conducted using `default` parameters in Weka for each model on a machine with `16 GB 2133 MHz LPDDR3` memory, and `Intel 2,8 GHz Intel Core i7 quad-core` processor. Results of these experiments can be found in Table 4.2. As we can see, oveall all the models except the NaiveBayes classifier showed good performance (more than 85% accuracy in both validation and testing phases). Specifically, Random Forest model stands out with a 91% accuracy in the testing phase, and up to 92% average accuracy in the validation phase. In Figure 4.6 we show the confusion matrix for the Random Forest. The confusion matrix shows predicted labels (by RF) against true labels (in the test set), i.e., the number of tp, tn, fp, fn. Thus, the Random Forest classifier shows a precision, computed as $\frac{(tp)}{(tp+fp)}$, of about 88,8%, while a recall, computed as $\frac{(tp)}{(tp+fn)}$, of 95,2%. We observe that some functional resources (15) have been classified as tracking. The reason is that those resources were from Content Delivery Networks and exibits similar HTTP traffic as for tracking ones because at the same time provides functionalities and analytics to the website [297].

Model	**Validation k=10**		**Validation k=5**		**Testing**
	Avg. accuracy	*Std. Error*	*Avg. accuracy*	*Std. Error*	*Accuracy*
NB	0,77	0,03	0,76	0,05	0,79
RF	**0,92**	0,01	**0,92**	0,01	**0,91**
MLP	0,89	0,02	0,88	0,02	0,87
SVM	0,88	0,02	0,87	0,03	0,86
J48	0,89	0,01	0,90	0,01	0,87

Table 4.2: Average (avg.) accuracy achieved by the machine learining models in the 10-fold (5-fold) cross-validation phase (*Validation*), and accuracy achieved in the testing phase on the test set (*Testing*). In **bold** the best result.

4.3.2 Discussion

In this section, we studied a part of threats users face on the Web, in particular the 3rd-party tracking to derive whether it is possible to distinguish between functional and tracking resources to provide privacy against threats online. Existing privacy tools mainly rely on blacklisting mechanisms which are limited because such lists must be manually updated and maintained. The exper-

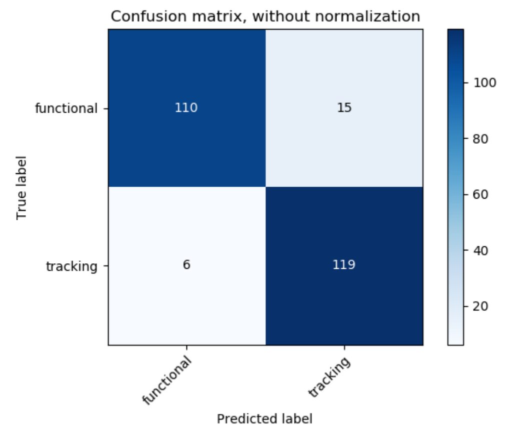


Figure 4.6: Confusion matrix for Random Forest model.

iments with ML to classify Web resources based on HTTP traffic features showed positive results, with the Random Forest classifier able to classify functional and tracking resources with an accuracy of 91%. Our result shows therefore a new perspective in embedding the use of ML in a privacy tool such as AdBlock or Ghostery, i.e., a client-side application. The idea is to develop a browser extension powered by ML which is capable to filter out tracking resources, and fill a customized blacklist accordingly. The blacklist will be customized because each user has its own browsing habit, therefore its blacklist will include only the threats the user is exposed to. In this way, we reduce the time needed for the ML computation for each Web resource that has already been processed by the tool. Such privacy tool can include also visual analytics techniques [298, 299, 300, 301] to give the user the right level of awareness while browsing, for instance by showing the graph of the HTTP requests performed (and blocked) during the browsing session.

4.4 Hybrid and lightweight detection of third party trackers on the Web. Design, Implementation and Evaluation

We provide here an advancement of what proposed in Section 4.3, experimenting the envisioned ML techniques on a relevant dataset.

The challenges we address in this section, concern from one hand, the size of the blacklists whose management represents a problem for performance (mainly in terms of system memory usage) and, on the other hand, the arduousness (human intervention) of keeping the blacklists up to date, in order to make them effective against the new aggregators entering the advertising landscape. The idea of relying on an automatic approach to classify 3rd-party resources with the final goal of avoiding the use of blacklists manually built and maintained is not new [266, 267, 268, 269]. However, although these works present different approaches for resource filtering, none of them provide a real countermeasure that deploy it, providing evidence of its effectiveness during "*online*" activities.

This section presents a *hybrid* and *lightweight* approach for 3rd-party resource filtering envisioning the use of *tiny* lists built according to the browsing habits of Internet users.

4.4.1 A ML approach for 3rd-party filtering

In this section, we describe the ML approach we designed to automatically detect malicious behaviors and filter out 3rd-party tracking requests while browsing the Web.

We want to emphasize that our goal is to design a ML approach that allow us to provide a valid and effective instrument to protect privacy without performance degradation respect to the existing privacy tools that exclusively rely on blacklisting mechanisms. The idea is to verify whether ML can be efficiently employed in the real time application of filtering unwanted content, without impacting on the user experience; moreover, we want also to overcome the problems of having manually update blacklists and reducing the overhead associated with the use of a high number of regular expressions to apply filtering mechanisms. Our blacklisting mechanism indeed, being personalized according to the real habits of a user, produces files of unwanted resources smaller than those used by the most popular privacy tools available in literature.

We analyzed different classifiers able to distinguish *malicious* (tracking) requests and *functional* requests. Therefore, our classification problem is a binary classification problem on Web resources where positive examples are tracking requests and negative examples are functional resources.

In the following, we describe in details the procedure used to design our ML-based approach. As we can see in Figs. 4.7 and 4.8, both for JavaScript programs and HTTP requests the procedure consists of three phases: *(1)* in the *Data Collection* phase we collected the Web resources (JavaScript programs and HTTP requests) necessary to build an initial raw dataset, *(2)* in the *Dataset Building* phase we used syntactic features (resp. Web traffic features) extracted from JavaScript programs (resp. HTTP requests) and collected previously in the raw dataset, and finally, *(3)* in the *Training and Testing* phase we tested different ML classification models, available in literature.

Data collection

To gather data, as did in other similar works [266], we leverage on the labeling provided by Ghostery and Disconnect, given their popularity and their constantly up-to-date databases. A detailed description of these privacy tools is provided in Section 4.2.2. We modified the source code of these tools in order to log information about the retrieved URL and its corresponding label ("*Blocked*" or "*Allowed*"). Moreover, we developed an auxiliary add-on for Google Chrome, named WatchHTTP, to log HTTP headers of each issued Web request and furthermore to indicate if that request is a first party one or a 3rd-party one. Finally, we developed a

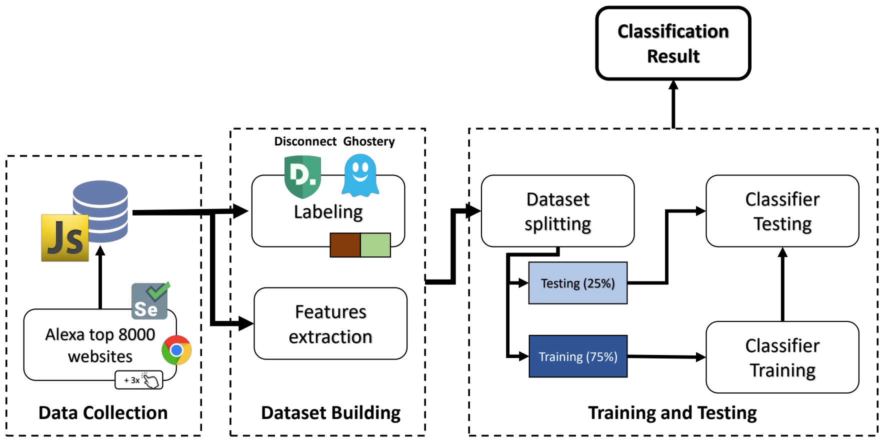


Figure 4.7: Procedure: Data Collection, Dataset Building, Training and Testing for JavaScript Programs resources.

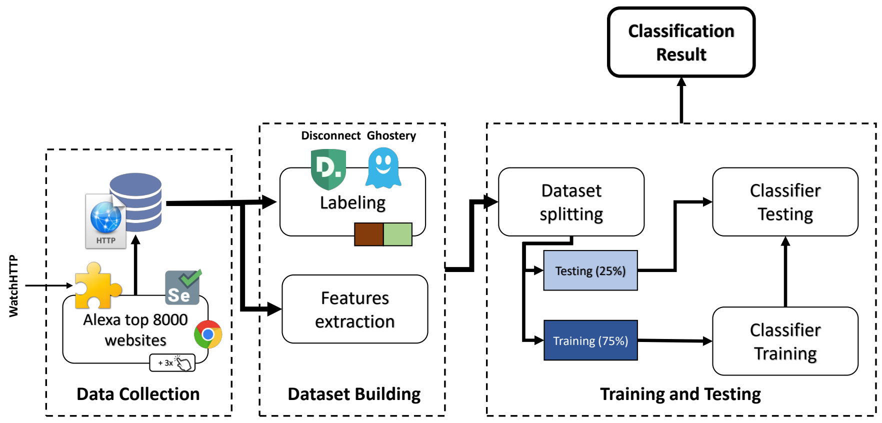


Figure 4.8: Procedure: Data Collection, Dataset Building, Training and Testing for HTTP Requests resources.

crawler based on the Selenium webdriver[6]. We crawled the Alexa top 8,000 sites, randomly following 3 links on each page, for a total amount of 32,000 downloaded websites. Broken links were next dropped out from the final list. Crawling has been performed

[6]https://www.seleniumhq.org/

in parallel, by envisioning three different instances of Selenium webdriver, two for Ghostery and Disconnect (to leverage on their labeling of JavaScript programs) and one for WatchHTTP (to log traffic features of HTTP requests).

Through this crawling process we were able to build two distinct datasets: *(1)* the JavaScript programs dataset, with 20,516 elements, and *(2)* the HTTP requests dataset composed of 263,447 elements. Just to give an idea of the current ecosystem of 3rd-party aggregators and of their pervasiveness, in Fig. 4.9 we show the top-20 aggregators found out in our analysis (identified through the labeling of Ghostery and Disconnect), while browsing the top-8000 Alexa websites. As we can see, in terms of extent, Google is the leading family with 35% of controlled aggregators worldwide.

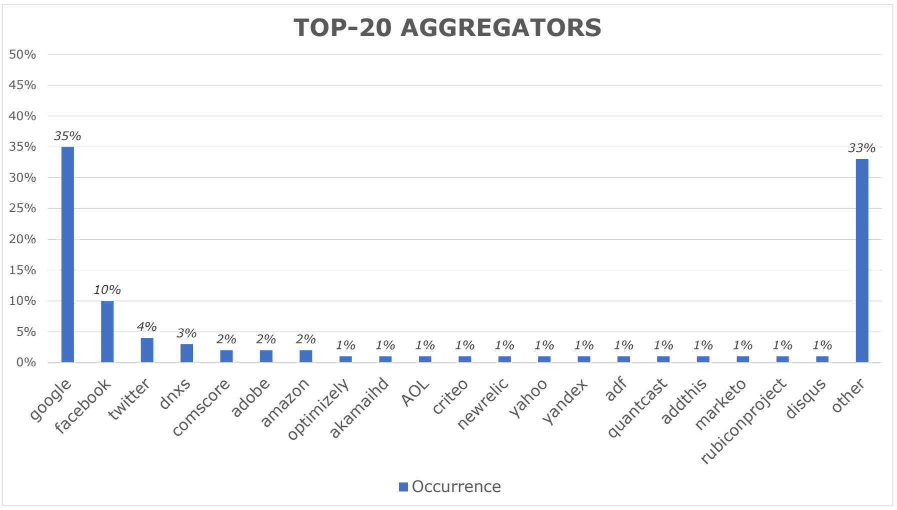


Figure 4.9: Distribution of the aggregators observed during the crawling process on 32,000 analyzed websites.

As explained in Section 4.2, Ghostery and Disconnect make use of blacklists containing a list of resources to be blocked. In our approach, each sample of the datasets has been marked as functional or malicious according to the labeling realized by Ghostery and Disconnect (what is in their blacklists). Specifically, we adopt the following strategy: *(1)* a sample has been marked as functional if both Ghostery and Disconnect *simultaneously* considered it as a functional resource (and therefore both allowed it), and conversely, *(2)* a sample has been marked as malicious *if one* of the two add-ons (or both) considered it as a tracking resource (and therefore they blocked it). By using both systems we were able to identify the larger number of resources taking also into account (not skipping) resources wrongly "*allowed*" because of, as an example, internal commercial agreements (Acceptable Ads programs). As an example the malicious JavaScript program `chartbeat_mab.js` has been blocked by Ghostery but not by Disconnect. Details about the reosurces identified in our datasets are shown in Table 4.3.

Label	**JavaScript Programs**	**HTTP Requests**
Functional	12,813	150,500
Malicious	7,703	112,947

Table 4.3: Number of resources identified as Functional and Malicious.

Dataset Building

Here we describe how we built the two datasets used for the classification problem defined in our approach. We first describe the dataset composed of JavaScript Programs and then the second one regarding to HTTP headers.

JavaScript Programs dataset The JavaScript features considered in our analysis have been identified exploiting results of several previous studies [268, 269, 302, 303]. In particular, from these studies, we identified 13 features, that we will describe in detail in the following. In Table 4.4 we show the rules as well as the JavaScript programs that match them.

Rule	Description	% of JS Malicious	% of JS Functional
R1	Margin setting	13.86	24.04
R2	Hidden element creation and web bugs	38.93	16.44
R3	Mouse and keyboard tracking	30.55	42.32
R4	Dynamic creation of `iFrame` and `script`	48.40	35.83
R5	Third party JavaScript	80.94	33.22
R6	Relief and analysis of user browser and user platform	64.01	23.05
R7	Fingerprinting elements creation	19.49	5.08
R8	Referer usage	26.30	9.24
R9	Retrieving information about hostname, pathname, and protocol used	85.82	28.05
R10	Using Math library	66.09	27.23
R11	Cookie manipulation	31.66	23.64
R12	High occurrence of detected malicious keywords	40.72	20.61
R13	High occurrence of detected functional keywords	63.50	82.90

Table 4.4: Rules applied for feature extraction and the labeled JavaScript programs that match them.

Rules *R1* and *R2* in Table 4.4 match malicious JavaScript programs that create panel and set margin to host advertising content [268] and that create hidden elements and web bugs; *R3* matches functional JavaScript programs that tracks keyboard and mouse to improve the user experience [268]; *R4* and *R7* refer to malicious JavaScript programs that create dynamic `script` and `iFrame` elements as well as fingerprinting elements (i.e., `canvas` or `ping` attributes [302]); *R8*, *R10*, and *R11* match respectively *(a)* JavaScript programs that use the `referer` header (that identifies the Web page linking or embedding the requested resource), *(b)* that exploit the Math library and *(c)* that access and manipulate cookies, with the common goal of tracking users. To better explain *R10*, the Math.random() function generates pseudo-random numbers in the range [0,1]; these numbers are strictly dependent on the platform on which the function is invoked. Researchers reported that it was possible to reverse engineer the value used to seed Math.random(), getting a unique token to identify and track users across different web sites [303]; *R6* refers to information about browsers and used platforms; *R12* and *R13* refer to keywords we found out with high occurrence in our datasets. Specifically, we calculated the term frequency and inverse document frequency (tf-idf) score [304] of keywords which do not map the other rules (i.e., *R1-R11*) and we selected only the ones with a high score for malicious and functional JavaScript programs, respectively. Finally, *R5* refers to the nature of a resource, that is, whether it is 3rd-party or not. As we can see from Table 4.4, in 80.94% of the times these resource are effectively malicious.

We want to emphasize that the rules we identified do not characterize a distinct separation between the two class (malicious or functional). For example, setting cookies, can be seen, at the same time, as a functional behavior (to maintain session) and as a malicious behavior (to track users). Moreover, features then calculated as the sum of the occurrence of keywords associated to a certain rule, by exploiting term frequency (tf). For instance, let be j a JavaScript program; if keywords $k1$ and $k2$ map the rule *Ri* then the feature is calculated as $tf(k1) + tf(k2)$ in j. Some examples of keywords we use and the rules they map them can be found in Table 4.5.

HTTP requests dataset Following the work done in [266, 267] we defined this dataset by analyzing features extracted by HTTP requests. Specifically, we identified 10 features of interest, some selected from previous works and other identified by us, that can be summarized as follows. The feature *F1* models the *number of parameters* set in the HTTP request, often associated with a malicious behavior. *F2* models the *cookie length*, since tracking cookies have an average length greater than 35 characters [267]. *F3* maps the *set cookie* field, a potential malicious behavior when the number of these cookies is high. *F4* represents the *cookie entropy*, and we selected it since we notice, that in our dataset, cookies set by trackers have a higher entropy; it is worth to note that Pri-

Rule	**Keywords**
R1	`margin-top`, `margin-left`, `margin-right`
R2	`visibility:hidden`, `display:none`, `width`
R3	`onmouseover`, `onkeyup`, `mouseout`
R4	`script`, `iFrame`, `text/javascript`
R5	–
R6	`useragent`, `navigator`, `appName`
R7	`canvas`, `ping`
R8	`referrer`
R9	`location.hostname`, `location.protocol`
R10	`Math.random`, `Math.getTime()`
R11	`Cookie`
R12	`encodeURI`, `encodeURIComponent`
R13	`textarea`, `createTextNode`, `left`

Table 4.5: Examples of rules. – means that there is not a keyword, the 3rd-party JavaScript program exists or not.

vacyBager, a tool that we describe in Section 4.2, grounded its ML approach on cookies' entropy. *F5* and *F6* represent feature that models the comparison between *sent bytes* and *received bytes* (used as an example to identify Web bugs [266]). *F7* models the *cookie lifetime*, since it tends to be much longer for non-tracking cookie [267]. We also identified an interesting feature, i.e., *F8*, that models the *daisy chaining behavior*. It defines a mechanism through which linking of several 3rd-party entities it may increase the chance of building detailed dossiers about users [251]. Daisy chaining is identified by examining the HTML body which includes an `iFrame` that automatically triggers a request to the first aggregator. The aggregator's response includes a JavaScript file which triggers a request to the second aggregator. The linkage between the aggregators can be seen via the Referer header. Finally, *F9* models the *type of a resource* while *F10* identifies if a resource is *3rd-party* or not.

Training and Testing

To identify a suitable ML approach, we compare the classification performance of different ML methods from scikit-learn[7], a free software ML library for the Python programming language. Specifically, we tested: Multilayer Perceptron (MLP) and Linear Discriminant Analysis (LDA) algorithms, Random forest, k-Nearest Neighbor, and Support Vector Machines (SVMs). The experiment was conducted on a machine with Intel i7-7700HQ CPU 2.80GHz processor and 16 GB memory. Moreover, we used an 75%-25% split for training and testing data. The performance, in terms of accuracy measures, of the tested models are depicted in Table 4.6. As we can see, the Random Forest classifier had the highest accuracy for both datasets. This is not a surprising result given its high performance when compared with other classifiers [305]. In our analysis Random Forest outperforms the other classifiers, i.e., MLP and SVM, since, conversely to them [306, 307, 308], it works better in presence of outliers. LDA instead achieves the worst performances due to the slightly unbalanced dataset: since the number of functional elements is greater than the number of malicious one (as shown in Table 4.3), LDA tends to (wrongly) classify the samples more frequently as functional (even if they are malicious). Finally, k-Nearest Neighbor achieves good performance when analyzing the HTTP Requests dataset due to the similarities among the samples: for instance, it is common to have samples with the same date in the `Expires` field of the HTTP requests (same *F7* feature); 3rd-party server often use tracking cookies (set-cookie header field) with a value fo the date very far in the future per tracking purposes. However, the MLP showed good results with an accuracy of about 87% for both datasets, by representing the second best classifier in our analysis.

[7] `https://scikit-learn.org/stable/`

Classification Method	**Avg. accuracy**	
	JavaScript programs	**HTTP Requests**
Random Forest	90.60%	97.00%
MLP	87.17%	87.06%
SVM	84.20%	86.76%
k-Nearest Neighbor	85.19%	95.20%
LDA	75.39%	74.50%

Table 4.6: Comparison of the average accuracy of the tested ML models.

4.4.2 *GuardOne*: A ML-based browser extension to protect privacy

In this section, we describe *GuardOne*, a prototype of a Google Chrome browser extension that combines the blacklisting technique with ML for the automatic classification and filtering of privacy-intrusive resources (i.e., malicious requests opposed to functional resources). To the best of our knowledge, *GuardOne* is a first prototype of a Google Chrome extension for protecting privacy on the Web that relies on ML techniques. The prototype, being a Chrome extension, has been developed in JavaScript programming language.

The idea and the implementation

As explained in Section 4.2.2, most systems for protecting Web privacy available in literature use the blacklisting technique to filter out unwanted content. They maintain lists of malicious resources to block or a set of rules expressed as regular expressions. One of the most significant problem of this type of technique lies in the fact that the size of these lists can be really huge, affecting negatively the performance of web browsing [265]. At the same time, the choice of Web resources to be included in the lists is left to experts that have to manually create or update them. The question we tried to respond is if it is possible to manage shorter lists and at the same time intelligently update them without affecting the user's browsing experience and eliminating the tedious and error-prone activity of the manual update.

The novel mechanism we propose in this chapter try to combine the well-known technique of filtering with ML techniques to automatize the overall process. Can we ensure effectiveness in filtering unwanted content by exploiting ML techniques implemented client-side, without affecting performance?

The first choice for implementing the proposed hybrid mechanism regards the ML model to be use for identifying malicious requests from functional ones. As shown in Section 4.4.1, we analyzed the performance of different ML models, by using the accuracy as comparison metric. However, when selecting the model to implement we taken into account other aspects in addition to accuracy. As we can see in Table 4.6, Random Forest and K-NN, when considering the HTTP Requests dataset, have the highest average accuracy scores. However, they present well-known memory space problems that make them less appropriate to be implemented as browser extension, that must filter out Web content in real time [309, 70]. For the JavaScript programs dataset instead Random Forest and MLP show the highest accuracy scores. It is worth to note that the memory usage is one of the most important aspect to take into consideration when providing functionalities on mobile devices [265, 310].

In summary, our main goal in this work is not identify the most suitable ML model to implement for our purposes but provide a privacy client-side tool, effective in terms of both application of filtering and performance, providing at the same time a new solution that avoid the tedious and error-prone activity of the manual definition of the rules to apply to block unwanted content. The model that best suits our needs is certainly MLP, which has no memory space problem (only the hidden layer neurons must be stored) which as we can see Table 4.6 has average accuracy scores of 87.17% and 87.06% for the JavaScript Programs and HTTP Requests datasets, respectively. Therefore, MLP has been implemented in *GuardOne* as ML model.

The workflow

An important characteristic of *GuardOne* is that it is a browser extension that customize its behaviors according to the user that is currently using it. Specifically, as we can see from Fig. 4.10, for each received request (1), if it is a new request ("Yes"), than it is passed to the ML module (3) to classify it as a malicious or as a functional resource. In the first case (4) the corresponding resource will be added in a blacklist ("Yes"), otherwise it will be added in a whitelist ("No"). Conversely, if the request has been received and processed in the past (2), then the extension check its presence in the blacklist, blocking (allowing) it in positive (negative) way.

This behavior allows the construction of customized lists according to the user's browsing habits: each user will have a own different configuration. The size of these blacklists is smaller than those used by the most popular privacy tools, since they take into account the habits of a single user, and have not been pre-built with all well-known unwanted resources.

Finally, although the automatic application of filtering, users can decide to take the control by simply accessing the user interface and changing the configurations.

A security perspective

In this section, we describe a typical adversary model and a type of attack that can be mitigated by *GuardOne*. As proposed in [311] we focus on a passive eavesdropper adversary, whose aim is to leverage 3rd-party HTTP tracking cookies for mass surveillance. We consider passive attacks for the following reasons: *(1)* passive attacks could be more powerful than generally realized, *(2)* active attacks, in general, begin with passive eavesdropping, and *(3)* passive attacks are easier to carry out, especially at large scale.

The passive eavesdropper adversary has the ability to inspect packet contents, e.g., HTTP cookies, with the goal to track a target user. One of the possible attacks is related to the authentication mechanisms. Most popular websites deploy he HTTPS protocol for authentication, but many Web pages reveal an already

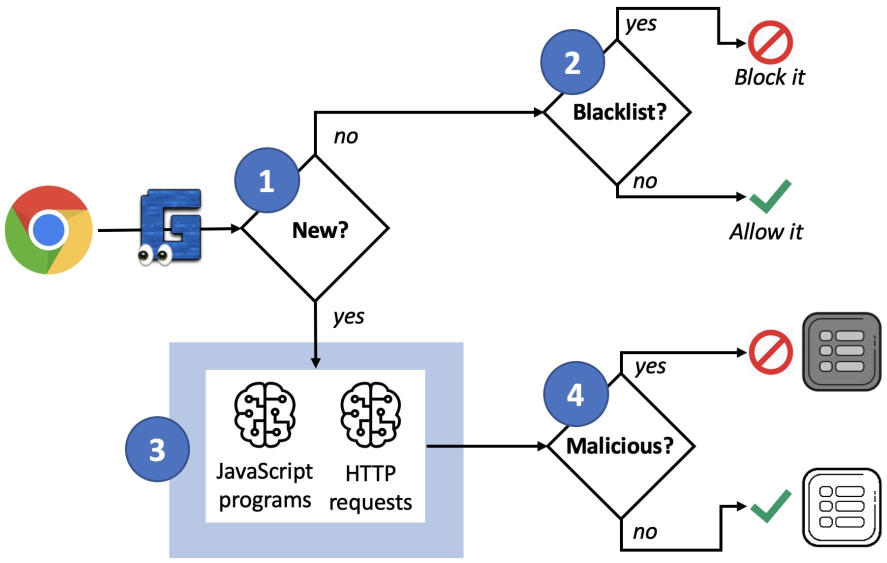


Figure 4.10: GuardOne management workflow.

logged in user's identity in plaintext. Thus, a passive eavesdropper can wiretap the user browsing session both to cluster together the Web pages visited, and to tie real-world identities to such clusters.

A typical attack scenario is described in [311]: an adversary can link visits to two different Web pages from the same user (even if the user's IP address changes) whether two Web pages embed the same malicious resource which tags the browser with a unique cookie. *GuardOne* sort out the root of the problem, by blocking and therefore avoiding to send cookies for resources classified as malicious ones. This means that the passive eavesdropper cannot wiretap a cookie that has not been ever sent.

4.4.3 Experimental study

Methodology

We present here the methodology employed for the experimental study, organized in three different class of experiments, we carried out to assess: *(1)* the performance of our privacy software prototype in terms of impact on user experience, *(2)* the effectiveness in terms of limiting personal information leakage, and finally, *(3)* the effectiveness in terms of filtering capabilities, by also analyzing the number of errors (i.e., false positive and false negative).

Tests were performed on different machines with identical environment to ensure that cookie, temporary files or any other aspect strictly related to the browser do not interfere with each other on the tests. Tests were performed on a PC Intel Core 2 Quad i7-2600 @3.40GHz with 8GB RAM and Linux x64 Ubuntu Operating System. The browser used for the experiments is Google Chrome, since currently this is the only implementation available for *GuardOne*.

Tools were analyzed regardless of how they are implemented and configured, trying to avoid, whenever possible, to modify the default settings. We run tests at the same time so that the influence of ad replacement is minimized. The results of each tool, or *PPTool*, have been compared individually to the results from running experiments without any tool installed, that we named *NoAddons*. Tests were performed on two workloads specifically built according to the analysis to perform.

Although the effectiveness of privacy tools has been studied in different works [251, 261, 262, 312], this is the first work that analyze approaches based on ML techniques in terms of accuracy (shown in Section 4.3.1) as well as effectiveness and performance of its software implementation (that we will describe in the next section).

***Test One*: Performance** The first workload we defined, named *"Top50CategoryWorkload"*, consists of the top-50 Websites from 14 Alexa categories, for a total amount of 700 Websites to be downloaded and analyzed. The categories include: *Arts, Business, Computers, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Science, Shopping, Society and Sports.*

Tests envisioned the automated browsing of this workload (through a module developed in our measurement framework), gathering information about HTTP header requests/responses, system statistics (percentages of RAM and CPU usage), total amounts and types of blocked resources. To measure the impact on user experience we performed tests to analyze the impact that the application of filtering techniques would have on the resource download time. We compared how the tested tools perform in terms of mean response times when applying the filtering capabilities by calculating the gain in terms of response time when 3rd-party objects are being removed from users' requests. Browser performance were analyzed by calculating both CPU and memory consumption during the browsing of resource from the *"Top50CategoryWorkload"*. Specifically, to collect this information different samplings are performed during the loading of a Web page; the corresponding data are saved to be analyzed next. Finally, data about processes (Google Chrome processes) were collected using different tools and Python libraries (i.e., *smem*, *psutils*, *subprocess*).

***Test Two*: Information Leakage** To analyze the broaden of the personal information leakage, we defined a specific workload, named *"InformationLeakageWorkload"*, composed of 11 Web pages, that is, facebook.com, yahoo.com, twitter.com, flickr.com, mail.google.com, tripadvisor.it, answers.yahoo.com, webmd.com, google.com, instgram.com, and youtube.com. We set up accounts with the corresponding first party sites rather than signing in via a 3rd-party account. We created a fake account for the Web sites that required a login access, by adding detailed information including full name, email address (required for all accounts), Date Of Birth (DOB), Social Security Number (SSN), zip code, home address, personal cellphone, school and general education information, sexual orientation, political and intellectual beliefs, general interests (music, movies, and travel). They represent the bits of private information that may be leaked towards 3rd-party sites. We created a log of typical interactions between the user and the sites. We included the following actions that may uniquely identify the users from (a) search terms [313], (b) browser habits, (c) preferences about music and movie, and (d) the structure of their social networks [314].

We used the following six types of online users' interactions:

1. *Account Login and Navigation* We logged in on all sites and analyzed information leakage due to 3d-party cookies. We also visited 4 or 5 embedded links per page, to reflect typical navigation of a user [315].

2. *Viewing/Editing Profile.* To reflect the most common actions performed by users on OSN we analyzed the following actions: viewing one's own profile and editing it (About link in the profile page, "Write About Yourself"), viewing 5 friend's profiles, writing on the "Timeline" of 2 of them.

3. *Searching the Web for Sensitive Terms.* We searched using google.com for terms in these seven categories: Health, Travel, Jobs, Race and Ethnicity, Religious beliefs, Philosophical and Political beliefs, Sexual orientation. For each search term we also navigated through the first 2 search result pages. We searched for two keywords across the 7 sensitive categories.

4. *Inputting content.* Since leakage of private information can occur when users input content on Web sites, we analyzed the following actions: post and reply to questions on forums, reply to dating messages, upload pictures.

5. *Like-ing content.* Leakage can occur through social plugins, such as Facebook Like Button, Google Plus Button, and so on. We analyzed the "Like" and the "Share" via Facebook.

6. *Daily actions.* This category includes some typical interactions such as viewing and sharing videos, searching for and booking trips, writing emails, interacting with social networks, chatting with friends.

We simulated these users' actions by automatizing them when browsing the resources contained in the *"InformationLeakage-Workload"*. Through a specific module of our measurement framework we gathered statistics about HTTP requests and responses (Headers and Body).

***Test Three*: Effectiveness** To assess which *PPTool* is most effective in terms of blocking unwanted content during Web browsing we performed two types of analysis:

- *quantitative*, to quantify the number of blocked resources when issuing requests from the *"Top50CategoryWorkload"*

- *qualitative*, to derive the number of false positives and negatives when applying filtering techniques. Here, we manually inspected *headers HTTP(s)*{*request, response*} of the HTTP requests issued for resources belonging to the *"Information-LeakageWorkload"*.

Specific modules implemented in our measurement framework were used to automatic browse pages from the two workloads and calculate all statistics about blocked (*"Crawling and performance"* module) resources and number of errors (*"Check Information leakage"* module).

The measurement framework (DMF)

To analyze and compare the browser extensions, we developed, *DMF*, a distributed modular measurement framework in Python. Specifically, we exploited the following APIs and tools: (1) the Selenium WebDriver for browser automation (part of the Selenium project); (2) the ChromeDriver, an open source for automated testing of webapps across many browsers, in terms of navigating webpages, JavaScript execution, and more; (3) Python libraries to extract various specific information during the simulation of the tests (i.e., the use of CPU and RAM for Google Chrome processes).

A separate instance of the framework is launched for each of the six *PPTools* to test, with an additional one to perform the experiment for the baseline conditions (i.e., *NoAddons*).

A high-level system overview of the framework is shown in Fig. 4.11. It consists of different modules, that we describe in detail in the following.

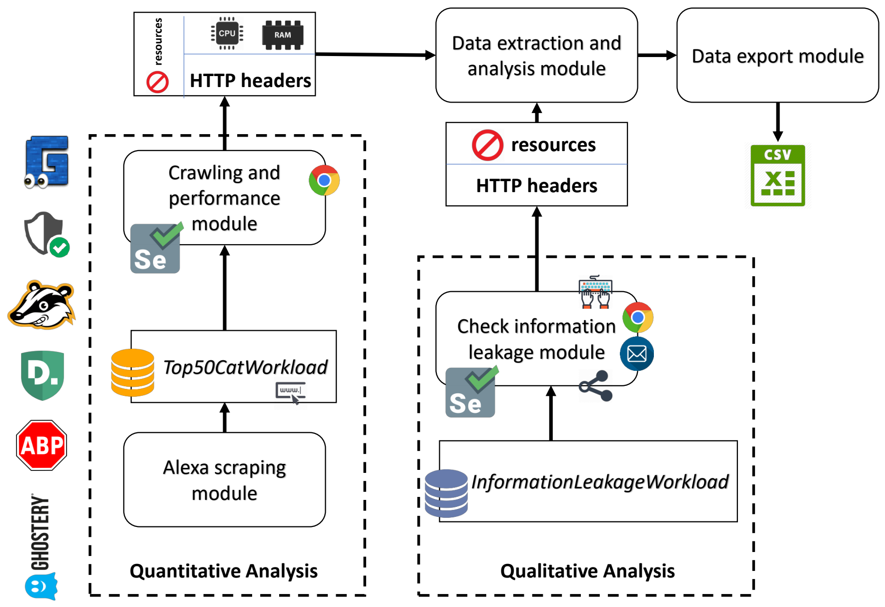


Figure 4.11: Overview of the measurement framework. It is responsible of task distribution, data collection handling, and finally, data analysis.

Alexa Scraping The main goal of this module is to extract the links from the Alexa[8] Web page to subsequently build the workloads used in our experiments. We built two different types of workloads according to the different type of experiment to perform (i.e., the *Top50CategoryWorkload* and the *InformationLeakageWorkload*).

Crawling and performance This module performs two main tasks: (1) crawling of the datasets of web pages (output of the

[8]https://www.alexa.com

Scraping module), by using Selenium WebDriver to automate the browsing of Web pages and to obtain specific information (i.e., Network Events from Google Chrome); (2) calculating of the system information, such as CPU percentage and RAM used by Google Chrome processes during the experiments. For both system statistics, several samples are taken and saved during Web pages browsing.

Check information leakage This module performs the analysis of the information leakage on the provided *InformationLeakage-Workload*. It uses Selenium WebDriver to automate tests, logging HTTP headers and saving both the HTML pages and JS codes.

Data extraction and analysis This module represents the core module of our framework. It takes in input data coming from the other modules and returns several information. It provide several functionalities, such as: parsing of event triggered during Web page browsing, calculation of statistics about HTTP requests/responses, statistics about blocked resources, statistics about CPU and Memory consumption, analysis of the log to quantify personal information leakage.

Data export This module is a simple CSV exporter to store data in CSV file format.

Results

The purpose of this analysis is to understand which *PPTool* is the most efficient in terms of hardware resource consumption (CPU and Memory Usage). Data are collected across several sampling during loading of Web pages. The *crawling and performance* module is in charge of performing this task while to analyze the results we exploit the framework's modules *Data Extraction and Analysis* and *Data Export*.

Browser performance results As we can see in Fig. 4.12, *Guardone* exhibits better results when compared with the other tools, with the only exception of *NoScript Lite*, that aggressively block resources and show therefore shows lower values for CPU consumption.

Conversely, *Guardone* is able to efficiently accomplish its filtering tasks without compromising the functioning of the rendered Web pages. In Fig. 4.13 we show the amount of memory allo-

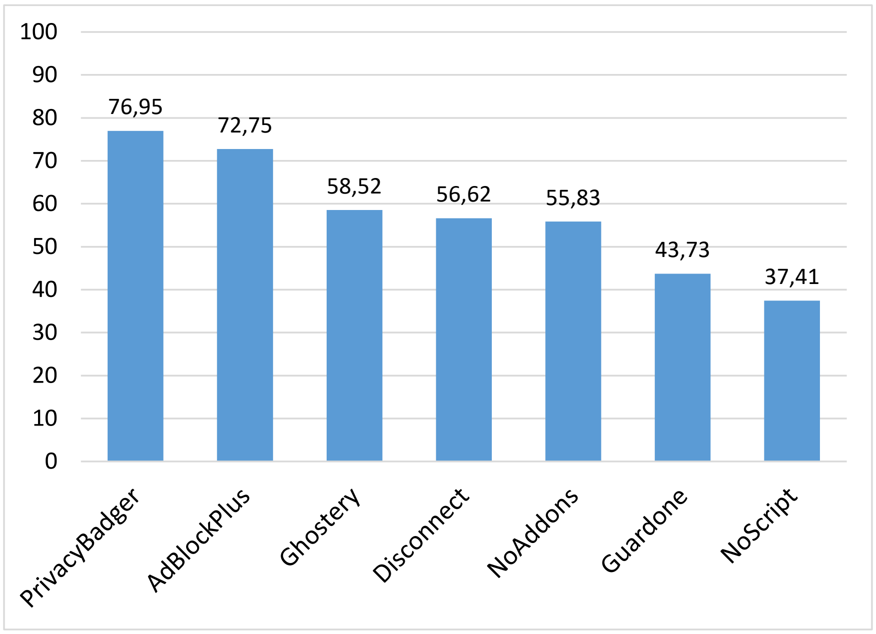


Figure 4.12: Percentages of CPU usage for each tested PPTool.

cated to Google Chrome processes' during the browsing of sites of *To50CategoryWorkload*. *AdBlockPlus* shows the highest value of memory consumption since it needs to load the subscription list. The best result is achieved by *GuardOne* which, exploiting ML techniques, does not rely on the use of *large* blacklists. Specifically, with our approach we are able to save about 58% of the memory used by *AdBlock Plus* in the same experiment. All others

PPTools, based on {*blacklisting, heuristics*} mechanisms, require more RAM, since they need to maintain the files with resources to block in the main memory.

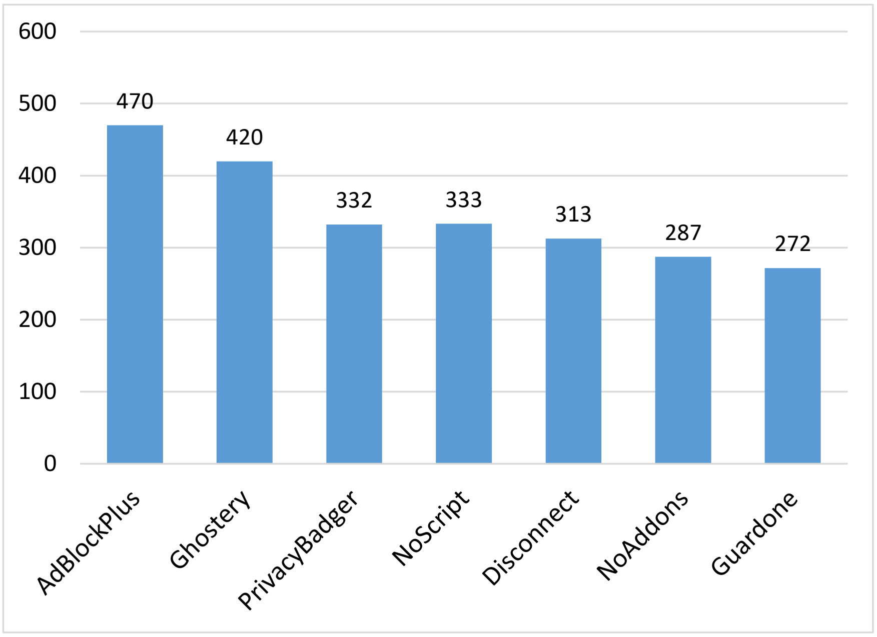


Figure 4.13: Memory (MB) usage of each tested PPTool.

Response time results We compared how the tested tools perform in terms of mean response times when applying the filtering capabilities on our data set (*50TopCategoryWorkload*). Specifically, we calculated the gain in terms of response time when all 3rd-party objects are being removed from users' requests. We computed the objects retrieved on a page when filtering is applied, against objects retrieved under normal or baseline conditions (i.e., the *NoAddons* experiment).

We have computed the CDFs of the response time for all PP-Tools and we have compared them with the CDFs of the response time when the Normal retrieval method is used (we call it the

NoAddons CDF). Fig. 4.14 shows the behavior of the tools when the own tracking techniques are applied on the stream of HTTP requests (*50TopCatWorkload*) against the behaviors of the Normal CDF.

Fig. 4.14 shows the time for download all resources of the Top50CatWorkload. For this experiment, the best response time, of about 12 seconds, is shown by *NoScript Lite*; although it may be see as a successful result, we have to emphasize that it is only due to the aggressive filtering of resources. *Ghostery*, *Disconnect* and *GuardOne* settle in the range 19-23 seconds, without compromising web pages functionalities (differently from *NoScript Lite*). So *GuardOne*, which uses ML techniques is comparable to *Ghostery* and *Disconnect* which conversely use blacklising techniques. Worse results are experienced by *AdBlock Plus* and *Privacy Badger*: around 26-28 seconds to load and render a Web page.

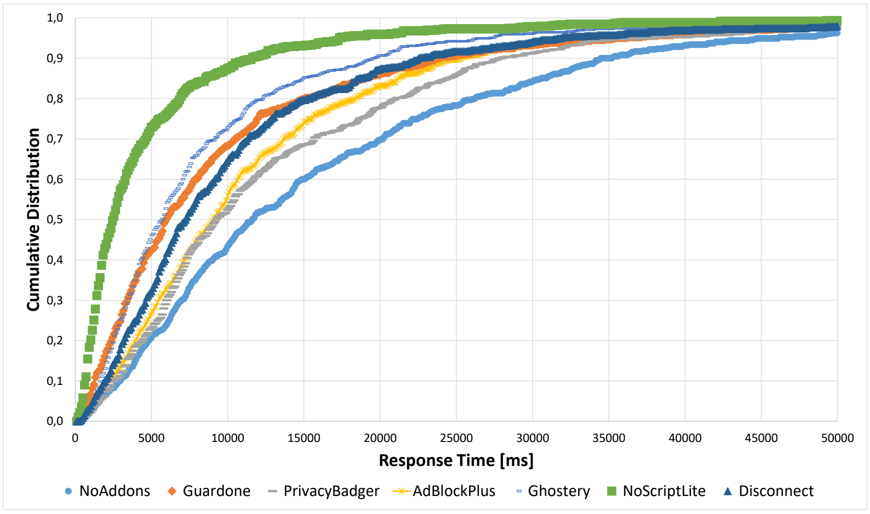


Figure 4.14: Cumulative distribution function of the response time for all PPTools.

We have also computed the CDFs of the byte session length for the PPTools and we have compared them with the CDFs of the byte session length when the Normal retrieval method is used (we call it the NoAddons CDF). As we can see from Fig. 4.15, this experiment confirms the previous results with *NoScript Lite* that cuts the highest number of resources, given its aggressive filtering behavior.

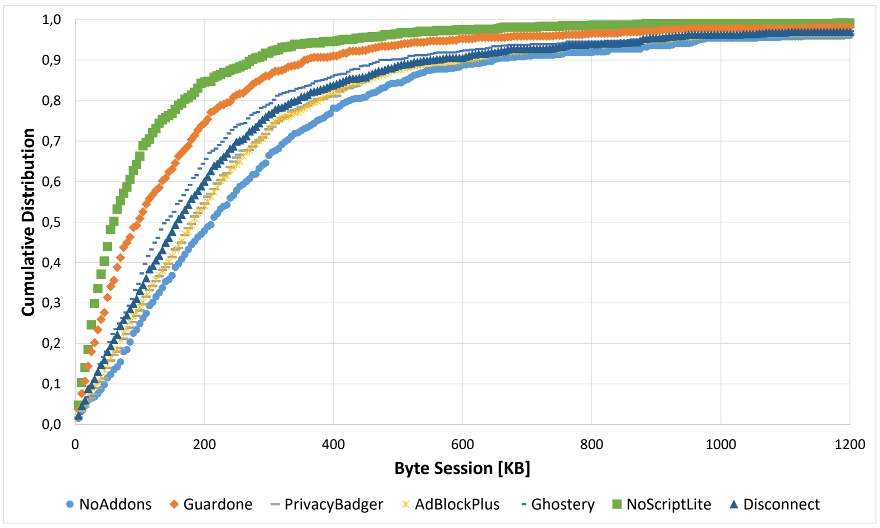


Figure 4.15: Cumulative distribution function of the byte session length for all PPTools.

Test Two: Information Leakage We describe here the results of the class of experiments whose main objective is to assess the effectiveness of the PPTools in terms of limiting the number of personal and sensitive information leaked towards 3rd-party sites. However, before describing the results of this class of experiments we have to illustrate some problems experienced by some tools.

- Given the blacklist-based approach used by No-Script Lite and its aggressive filtering, we were not able to get statistics for some sites. These sites were marked with the initials "*na*" in the tables presenting results of the analysis.

- To allow the browsing of some sites (Facebook, Twitter, Yahoo! and TripAdvisor we had to foresee a mini-whitelist to include content distribution networks (CDNs) for stylesheet files and images, marked as malicious by GuardOne. These servers were *connect.facebook.com*, *twimg.com*, *yimg.com*, *tacdn.com*.

- Ghostery blocked the analysis of Instragram, since it marked as malicious the associated cookie. This site was marked with the initials "*na*" in the tables presenting results of the analysis.

The first interesting result is about the low percentage of leakage for the "Personal Identifiable Information" (PII) category (only 0.22% of the total amount of the leaked bits). In Table 4.7, the column with heading "*NoAddons*" shows results for all the categories previously described. Specifically, as we can see, the worst result is about the Political Belief category with 45% of leakage (towards 163 different 3rd-party domains), and immediately thereafter the Travel Search category with a leakage of about 17% (towards 88 different 3rd-party domains).

Both Tables 4.7 and 4.8 show that when using a PPTool the leakage is reduced till 96% on average (specifically when using Ghostery). PPTools such as *GuardOne*, Disconnect, PrivacyBadger and AdBlockPlus, show a reduction of 81%, 76% , 54% and 44%, respectively. No-Script Lite deserves a separate comment. As we can see, one could think that it is the most effective tool given the drastic and desirable reduction of the information leakage. Conversely, most of the websites were not accessed given its filtering mechanisms.

GuardOne, a ML-based system, is ranked second in this leakage reduction ranking: once again automatic techniques are comparable with those based on the use of blacklists. In particular, for the Browser Fingerprinting category we have a reduction till 99% with Ghostery and 97% with *GuardOne*. The worst result is achieved by AdBlock Plus and Privacy Badger with reduction of 58% and 61%, respectively. Finally, observing the Health Searches

Leak type	**NoAddons**		**GuardOne**		**AdBlockPlus**	
	#Head=15706		#Head=9728		#Head=11774	
	Leak	Domain	Leak	Domain	Leak	Domain
PII	6	2	0	0	1	1
Browser Fingerprint	379	38	12	5	160	19
Travel Search	454	88	67	24	203	50
Job Search	39	8	6	2	39	8
Religious Beliefs	52	25	5	1	37	18
Political Beliefs	1203	163	109	24	501	75
Sexual Orientation	320	58	77	14	203	42
Health Search	207	32	20	7	196	33
General Searches	11	3	8	1	4	2

Table 4.7: Information leakage results for GuardOne and AdBlock Plus. NoAddons refers to the browsing without any PPTool installed. PII stands for Personally Identifiable Information.

category, either *GuardOne* and Ghostery show 90% of leakage reduction. The worst result is experienced, for the Health Searches category, by AdBlock Plus with a poor 5% of leakage reduction.

Test Three: Effectiveness Results of this analysis showed which *PPTool* was most effective in terms of filtering of unwanted content (3rd-party requests). As described in the previous section, we performed both a quantitative and qualitative analysis, whose results are presented in the following.

Results of the quantitative (conducted on *"Top50CategoryWorkload"*) are shown in Fig. 4.16. As we can see, results of this analysis place PPTools in three different sets. The first one includes *NoScript Lite* that indeed shows the largest blocking percentage. This filtering is not always correct (not all content is unwanted content), but it is simply due to the application of its strict set of filtering rules. The second set includes *GuardOne*, *Ghostery* and *Disconnect*, the first one applying a hybrid approach for filtering, the last two

Leak type	**PrivacyBadger**		**Ghostery**		**Disconnect**		**NoScriptLite**	
	#Head=12750		#Head=10315		#Head=9548		#Head=3651	
	Leak	Domain	Leak	Domain	Leak	Domain	Leak	Domain
P.I.I.	1	1	0	0	0	0	0	0
Browser F.	148	19	2	2	21	10	7	2
Travel Search	245	64	9	4	16	11	0	0
Job Search	23	4	4	1	7	2	0	0
Religious B.	27	14	3	2	27	17	2	2
Political B.	514	96	68	21	254	68	8	5
Sexual Orien.	16	5	6	2	47	16	1	1
Health Search	94	22	20	8	38	15	3	2
Other Search	12	3	0	0	10	2	1	1

Table 4.8: Information leakage results for PrivacyBadger, Ghostery, Disconnect and NoScript Lite.

a Blacklisting-based approach. Finally, the last set includes *AdBlock Plus* (blacklsting-based approach) and *Privacy Badger* (heuristics-based approach).

A result of this analysis is that *GuardOne* blocked a high percentage of a specific type of resources, that is, CSS files, and images. We deeply inspected the log files produced during the experiments, discovering that these (needed) resources have been blocked only because hosted by a special type of 3rd-party entities, that is Content Distribution Networks (CDNs). CDN are used to improve the user experience when browsing the Web. The AI implemented in our prototype correctly categorized them as malicious given their role of 3rd-party entity.

In Table 4.9 we show the amount of FPs and FNs found out when manually analyzing the resources from the "Information-LeakageWorkload" for each *PPTool*. We have to emphasize that we were not able to obtain complete results for Ghostery (just for one site the information are missing) and No-Script Lite (50% of Websites were unusable), and therefore we marked the corresponding information in the table with *"na"*.

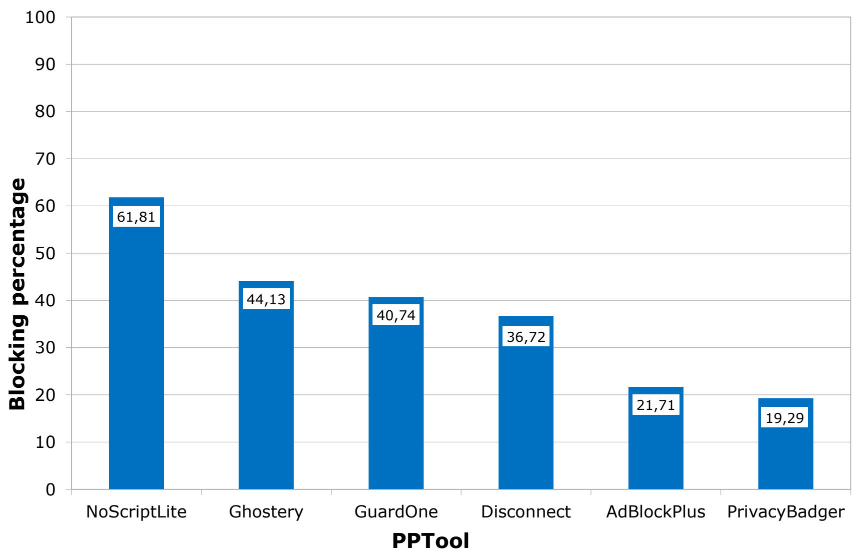


Figure 4.16: Effectiveness analysis: percentages of unwanted content for all PPTools.

As we can see, *GuardOne* exhibits the lower number of FNs whereas Disconnect results to be as the worst one. Conversely, *GuardOne* exhibits the highest number of FP. This behavior, has anticipated in the previous analysis is due to the fact that some resources, served by CDN, have been marked as malicious by our approach and the filtered out. All other PPTols exhibit a similar behavior except for No-Script Lite, which confirms the worst result even for this type of analysis.

4.5 Related work

In the recent past, there have been several approaches for detecting tracking behaviors and many works on privacy implications of Web tracking [271, 302, 316, 317]. Moreover, the idea of relying on an automatic approach to classify 3rd-party resources with the final goal of avoiding the use of blacklists manually built and maintained is not new [266, 267, 268, 269]. However, although these works

	GuardOne	AdBlockPlus	PrivacyBadger	Ghostery	Disconnect	NoScriptLite
	FP/FN	FP/FN	FP/FN	FP/FN	FP/FN	FP/FN
mail.google.com	0/0	0/0	0/0	0/0	0/0	0/0
yahoo.com	0/2	1/0	0/3	0/3	0/3	na/na
facebook.com	0/0	1/0	0/0	0/0	0/0	0/0
twitter.com	0/0	0/0	0/0	0/0	0/0	0/0
instagram.com	0/0	0/0	0/0	na/na	0/0	na/na
flickr.com	0/2	0/2	0/2	0/1	0/2	na/na
monster.com	0/10	1/13	0/15	0/9	0/15	na/na
tripadvisor.it	3/6	0/10	0/10	1/2	0/10	na/na
webmd.com	2/3	0/8	0/13	1/4	0/14	na/na
amazon.com	0/1	0/1	0/1	0/1	0/1	0/1
google.com	31/7	1/27	3/20	0/21	3/35	18/7
youtube.com	0/0	0/0	0/0	0/0	0/2	0/2
TOTAL	**36/31**	**4/61**	**3/73**	**2/41**	**3/82**	**18/10**

Table 4.9: Effectiveness analysis: false positives and false negatives for all PPTools. Columns with *"na"* mean that it was not possible to browse the corresponding website.

present different approaches for resource filtering, none of them provide a real countermeasure that deploy it, providing evidence of its effectiveness during "*online*" activities.

Among these works, some focused on advertisement content [318, 319, 320] or more in depth on tailoring the advertisement on the web [321, 322, 323, 324, 325, 326, 327]. Specifically, in [318] authors, for the first time, employed ML techniques to block online advertisements, using the C4.5 classification scheme. In [319] authors trained a classifier for detecting advertisements being loaded via JavaScript code with the features extracted using a static program analysis. In this study, they manually labeled advertisement-related JavaScript code and other JavaScript code of 339 websites by visiting them using Firebug[9]. Authors in [320] presented a technique for detecting advertisement resources exploiting the k-nearest neighbors classification leveraging on the EasyList, while in [328], authors extended their analysis to all Web trackers (not only advertisement) proposing a Web tracker detection and blacklist generation based on a temporal link analysis. They found out that their system was able to classify sus-

[9]https://addons.mozilla.org/sl/firefox/addon/firebug/

picious sites with high accuracy and detect new ads which might normally necessitate human intervention in the form of additional hand written filters. More recently, authors in [325] proposed a web-browser extension, which enables users to configure their own access policies to enforce a smart blocking over advertising platforms. The proposed approach exploits two methods: one that try to ascertain the extent to which user-browsing interests are exploited by advertising platforms, and another one that try to detect the uniqueness of the user profiles by means of a trusted server that retrieves and compares users' profiles. The data gathered through these two methods together with the policies defined by the users allow the web-browser extension to decide whether to block or not certain advertisements. Authors evolved the web-browser extension in [326] by adding a functionality that blocks or allows advertisements depending on the economic compensation provided by the adverting platforms to the users. A very recent example is in [327] where authors proposed a system, implementing simulated browsing sessions as a privacy-preserving measure, that conciliates users' privacy during web surfing and the advertising business. Specifically, their system empowers the user in the protection of her privacy by allowing her to define her privacy requirements, that is, which users interests should be hidden from the advertising platforms and which ones can be revealed. Then, the system selectively blocks or bypasses tracking on the browsed web sites according to their content and the privacy requirements. Their system has not been evaluated in real scenarios.

The work proposed in [266] is very similar to ours, because the authors propose a ML approach to detect tracking JavaScript codes. However, they use HTTP traffic traces instead of code analysis and their approach requires collaboration between different network nodes and services to gather traffic traces to apply ML algorithms on them. In comparison, our approach is used via a client-side browser extension where the ML algorithm is applied on the Web pages directly.

In [269], authors propose a system named TrackerDetector to detect 3rd-party trackers automatically. They focused on

JavaScript programs (33,366 elements), crawling them from 6,441 Web pages, and exploiting the Ghostery's labeling. An initial tracker set of 1,000 elements was manually labeled thanks to defined rules and a BFTree classifier was trained incrementally on the dataset. They individuated 717 Boolean features representing JavaScript API calls in the JS code, extracted via a JavaScript parser. Parsing a JavaScript files on-the-fly into the browser is high expensive in terms of performance, and relatively to this aspect we substantially differ from this work. Furthermore they only define the approach without providing any software implementation.

In [268], authors downloaded JavaScript files from 100 websites building a dataset of 2,612 files. They performed a manual labeling of the dataset, through 12 defined behavior rules. They tried different approaches among which there is a syntactic model and a semantic model. The syntactic model uses the top-200 JavaScript code lines ranked with tf-idf index as features, while the semantic model uses about 200 PDG (Program Dependency Graph) 7-grams features. Both approaches are exploited using a one-class learning classifier, specifically a One Class SVM and a Positive and Unlabeled Learning [329] [330]. With the semantic approach they reached about 99% of accuracy but this technique is incredibly resource-consuming to be deployed in a Web browser, but this clearly was not their purpose. Moreover they only focused on JavaScript codes, as also in [269], while we faced many other privacy threats by analyzing Web traffic features extracted from HTTP requests.

In [331] authors designed an effective Web browsing history anonymization scheme to protect users' privacy while retaining the utility of their Web browsing history. The proposed model exploits k-means [332]. It pollutes users' Web browsing history by automatically inferring how many and what links should be added to the history while addressing the utility-privacy trade-off challenge. Authors have then validated the quality of the manipulated Web browsing history and examined the robustness of the proposed approach for user privacy protection. As expressed by authors, the proposed model can not be personalized according to user needs and has not been tested on real world data.

Authors in [333] proposed a machine-learning framework for supporting intelligent web phishing detection and analysis. In particular they used decision tree algorithms for detecting whether a Web site is able to perform phishing activities. In the positive case, the system classifies it as web-phishing site. The adopted ML model exploits URL-based features such as URL length and domain age. The experimental evaluation confirms the benefits of applying ML methods to the well-known web-phishing detection problem. The model, as stated by authors, do not analyze the JavaScript code to detect the phishing activity, and, furthermore, has not been implemented in a web-browser extension or tested with real data. Finally, in Table 4.10 we show the main differences between our work and other relevant works proposed in literature. The comparison is made in terms of the following metrics: *(1)* types of threats addressed, *(2)* whether the work exploits ML techniques, *(3)* type of protection provided, *(4)* availability of a software implementation and, *(5)* whether "In vivo" experiments have been performed.

4.6 Conclusion

Although much attention has been devoted to studying the phenomenon of 3rd-party tracking to protect privacy, there exist no holistic solution that simultaneously result user-friendly, effective and efficient. To the best of our knowledge, all existing practical solution heavily rely on the use of static blacklist to remove unwanted content. Human efforts are required to allow blacklists to be maintained and constantly updated. Not least, the performance aspect appears to be a critical issue, given the high consumption of system resources, mainly when considering mobile devices.

To overcome these limitations, we developed a hybrid mechanism, exploiting a ML-based approach, to classify 3rd-party trackers. We were able to automatically learn currently known trackers

Reference	Threats	Machine learning	Customized	Software Implementation	"In vivo" experiments
[320]	Advertisement	✓	✓	✗	✗
[325]	Advertisement	✗	✓	✓	✓
[327]	Advertisement	✗	✓	✗	✗
[266]	JS programs	✓	✗	✗	✓
[268]	JS programs	✓	✓	✗	✗
[269]	JS programs	✓	✓	✗	✗
[328]	All web trackers	✗	✗	✗	✗
[334]	All web trackers	✓	✗	✗	✗
Our work	All web trackers	✓	✓	✓	✓

Table 4.10: Detailed comparison of our work with other relevant works available in literature. Our work takes into consideration all types of Web tracking, propose a ML-based Google Chrome extension that, in a customized way, protect privacy of Internet users. The effectiveness and the performance of the Chrome extension have been investigated on real data, through "In vivo" experiments.

with a high accuracy and up to 90% and 97% for JavaScript programs and HTTP requests, respectively. Our hybrid mechanism involved the use of small lists since they are built according to the user interest profile. We deployed our hybrid mechanism in a browser Chrome extension and we performed an exhaustive evaluation of performance and effectiveness of our software prototype showing that ML-based techniques can be employed client-side as solutions into the browser. Indeed, we showed that *GuardOne* is able to filter out malicious resources from users' requests with high efficacy and low impact on user performance, representing therefore a promising solution in the direction of automated approaches implemented client-side.

In our experiments we found out that, when analyzing effectiveness, the errors were mainly due to the large use of Content Distribution Networks in the modern HTTP interactions, by representing a serious issue for our approach. A correct classification requires information about the presence of a CDN and of its behavior, and this issue is currently under investigation. Moreover, a limitation of our work is that we built our dataset by leveraging on the labeling provided by Ghostery and Disconnect, and therefore we trusted them about their behaviors. A further analysis may be performed by using manually labeled dataset, with an high number of experts, and comparing the two different approaches.

However, given the encouraging preliminary results about the use of automated approaches to enable accurate and efficient Web tracking countermeasure when deployed as a client-side solution, we are currently working on how to mature the prototype presented here for its widespread usage through the chrome web store[10]. We are also working in the direction of improving accuracy, by analyzing more sophisticated classifiers that can be efficiently deployed into Web browsers. Another interesting direction could be the study of the accuracy when multiple classifiers are used, one for each type of Web resource to classify. Finally, once translated our prototype in the final software solution, an evaluation study, with a large and diversified number of participants, could be planned to assess usability and overall user satisfaction.

[10]https://chrome.google.com/webstore/category/extensions

Chapter 5

ML and privacy in practice: awareness

Chapter Highlights

- Laying the foundation of a privacy awareness system which warns users when they are about to disclose personal or private data, in text format, online;
- Definition of a ML approach to classify the text topic and sensitiveness of the content;
- Definition of a ML approach to learn users' attitudes towards privacy and customize warnings;
- Development of such ML approaches into *Knoxly*, a prototype Google Chrome extension which adopts *nudge*-based solution to promote users' awareness and foster privacy-oriented behaviors;
- Real-world evaluation of *Knoxly* in terms of effectiveness and efficiency finding that it does not "weights" the user experience and comparable with solution available in literature;
- Classification according to the taxonomy depicted in Chapter 2: **{ Type of work**: Tool, **Threat**: Dis, **ML Method**: RF, **Type of protection**: Nudge **}**

5.1 Introduction

In the social media era, people all around the world spend an ever-increasing time in socializing. In some countries, users are connected on social up to four hours per day [335]. During this time, they post content, send messages, react to someone else post, and so on. All these activities contribute to the dissemination of personal information, but there are cases where the information disclosed concerns other people. Users trust that much in the social media they use, that they are willing to disseminate significant amounts of private data without think twice [336, 337], revealing their whole life and lifestyle to followers. Actually, the audience of the social network is significantly larger than users' expectations (i.e., just followers and friends), which includes advertisers, analytics, search engine bots, and so on. The side effects of posts is completely neglected. As much private information is hidden in the text format postings, human stalkers or bots could collect old posts, which are delicate puzzle pieces to recreate the whole picture of the users' identities and lives [338]. To a certain extent, this is something already done because the digital traces that we leave on social media are like "footprints" that describe our behavior, both individually and as groups [339], precious food for all the Big Data economy. For instance, the infamous Cambridge Analytica scandal[1] was about personal data shared on Facebook sold for political campaigns. On the other hand, there is the real risk that all such data can be lost or stolen: in the last two years only, billions of data records have been stolen or made available due to several data breaches [340].

Researchers, developers, and regulators have dedicated much effort on developing privacy safeguards in digital settings. We can mention the well-known Do Not Track system (whose effectiveness has been proved to be poor [341]), the General Data Protection Regulation in 2018[2] but also tools for the end-user which aim

[1]https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html

[2]https://ec.europa.eu/info/law/law-topic/data-protection/

at protecting his/her privacy like the famous Ghostery [279] and ADBlockPlus [342] or the software proposed in [343, 344, 219]. On the one hand, the regulations trace guidelines for future developments; on the other hand, these systems mainly focus on protecting users from the tracking practice, enforcing the constraints without the user intervention. But this represents just one of the problems; there is another crucial component in protecting privacy beyond the enforcement of strategies to protect identities (e.g., see [345]), which is give users "the ability to control the dissemination of sensitive information" [21]. This draws from the concept of nudge [346] which has been applied since 2008 for several regulatory goals [10] protecting the rights of citizens in a "softer form", exploiting alerts and warning to let the targeted individual making the relevant decision [38, 32] (see Section 1.3).

Objectives Against this backdrop, we argue there is a need for a mechanism to distinguish potentially sensitive/private messages/-posts before they are sent. In particular, we want to lay the foundation of a system capable of: (i) detecting sensitive or personal data in a text, (ii) understanding the semantics of a text written in natural language, (iii) detecting potential privacy risks (i.e., dissemination of sensitive information), (iv) adapting to the user preferences. Therefore, the idea is to provide privacy awareness, already proved to be crucial to improve and guide people's behavior online [25]. At the same time, the system should be: (i) lightweight (i.e., no high computation needed in order to preserve users devices), (ii) fast, (iii) effective (i.e., capable of not missing sensitive contents).

The proposed strategy In this work, we present the design and the implementation of a system primarily based on ML and sentence embedding techniques, which is capable of classifying sensitive contents of a message based on what is the subject topic and adapting to the users' attitudes towards privacy. The conceived

`eu-data-protection-rules_en`

system encompasses four modules: (i) *Keyword module* - consisting of lexical mechanisms to detect sensitive or personal data in a test, (ii) *Topic module* - consisting in a ML model which distinguishes the text's topic, (iii) *Sensitiveness module* - consisting in a ML model assessing the privacy risk of a text based on the topic, (iv) *Customized module* - consisting in an online classification algorithm to learn users privacy attitudes. Such a system has been built upon the crawling of publicly available datasets on Kaggle, and, when not possible, using Twitter data. It has been embedded into *Knoxly*, a prototype Google Chrome extension, which provides the users with *privacy awareness* mechanisms triggering once a text written in natural language is about to be disseminated online (an overview of the proposed system is available in Figure 5.1). The awareness is delivered through the use of natural visualization techniques [347, 348]. The tool has undergone an evaluation phase to assess its efficiency and effectiveness when immersed in a realistic simulated environment via Selenium WebDriver [289].

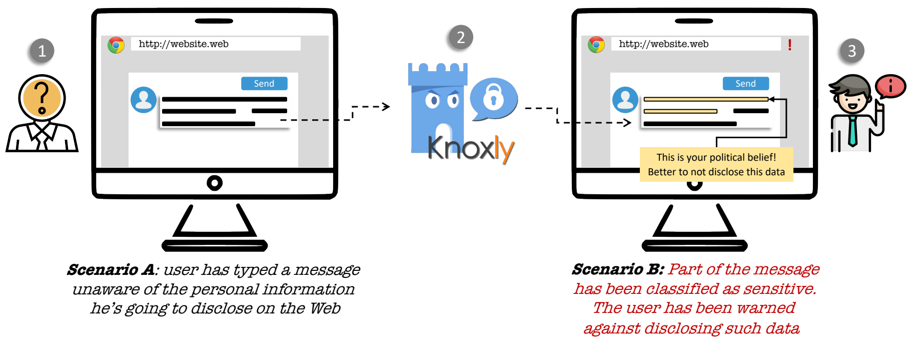

1
http://website.web
Send
2
Knoxly
http://website.web
Send
This is your political belief!
Better to not disclose this data
3
Scenario A: user has typed a message unaware of the personal information he's going to disclose on the Web
Scenario B: Part of the message has been classified as sensitive. The user has been warned against disclosing such data

Figure 5.1: An overview of the proposed system to nudge users providing privacy awareness. *(1)* The user has typed a message unaware of the information he is about to disclose online, *(2) Knoxly* analyzes the message, *(3) Knoxly* nudges the user providing awareness about the personal data within the message; now the user is aware of the risks he/she is facing.

Main contributions Our work, thus, presents an innovative approach to privacy awareness online. The main points can be summarized as follows:

- To design and implement a ML approach for classifying the topic of a text among the most studied in the privacy literature, i.e., politics, "health", "job", "travel", "racism", "religion", and "sexual orientation" to which we added a "general" topic in order to create noise and a more comprehensive model (i.e., *Topic module*);

- To design and implement a ML approach for classifying the sensitiveness of the content between sensitive and non-sensitive on the basis of the message's topic (i.e., *Sensitiveness module*);

- To design and implement a mechanism to learn online the users attitudes towards privacy and consequently adapt the sensitiveness level of certain contents (i.e., *Customized module*);

- To provide the embedding of the envisioned solutions into *Knoxly*, a prototype Google Chrome extension which is, to the best of our knowledge, the first tool for the end-user providing him with the ability to control the dissemination of sensitive information; the tool performs this task by means of privacy awareness techniques based on a natural User Interface;

- To provide an evaluation of *Knoxly* both in terms of efficiency (i.e., resources allocated) and effectiveness (i.e., quality of classification performed and so alerts for the end-user). This is one of the first solutions of this kind tested "in vivo" [349] (with users' real browsing sessions) to analyze both effectiveness and impact on the user's experience;

- The proposed solution can also be used by non-human users such as social media chatbots. For instance, Microsoft's

Twitter bot, Tay, started to deliver racist and hateful content soon after it was launched in 2016. Our solution could stop this unwanted behavior.

Chapter structure The rest of the chapter is organized as follows. In Section 5.2 we introduce the background needed to understand the rest of the work; in Section 5.3 we describe the methodology adopted for the design and implementation of our system capable of classifying sensitive contents of a message based on what is the subject topic and adapting to the users attitudes towards privacy; Section 5.4 presents and details *Knoxly* extension for Google Chrome with the workflow and the fronted; in Section 5.5 we show the experiments performed with *Knoxly* and the results so far obtained; in Section 5.6 we detail the most relevant studies showing contact points with our work; lastly, Section 5.7 is devoted to discussion about limitations, final remarks, and future works.

5.2 Preliminaries

This section provides basic information useful to understand the remaining part of the work. In particular, we provide notions on data and their sensitiveness (Section 5.2.1), the adversary model (Section 5.2.2), advanced techniques for natural language processing (Section **??**), and the ML methods used and how we measured their performance (Section **??**).

5.2.1 Sensitive and personal data

During the years, and in step with the development of Web, ever more data have been disseminated online by citizens: photos, locations, private events of personal life, and so on. Such data have turned to be precious in feeding an entire economy based on data where the more personal and specific they are the more money companies can earn [350]. In contrast with this kind of economy based on intruding the citizens' privacy, regulators and researchers have proposed several solution to safeguard people's private lives. Apart from results obtained from a privacy and security technical point of view (e.g., see [351, 352, 219]), such contributions helped in the creation of a taxonomy of such data. In this section, we briefly sketch the taxonomy of data with regards to privacy.

Personally Identifiable Information (PII) This category has been defined in [353]. PII refer to "information which can be used to distinguish or trace an individual's identity either alone or when combined with other information that is linkable to a specific individual". That is, email address, telephone number, date of birth, personal photo and so on.

Quasi-identifier (QI) This category has been introduced in [354]. QI refer to data that analyzed as singleton does not disclose a person identity, but when analyzed in set are capable to uniquely identify a person identity. It was proved that with just ZIP code, date of birth, and gender authors were capable to individuate up to 87% of American citizens [354].

Sensitive Data (SD) The recently introduced General Data Protection Regulation in the European Union[3] has highlighted which are the data that fall in this category, that is: "personal data revealing racism or ethnic origin, political opinions, religious or philosophical beliefs; trade-union membership; genetic data, biometric data processed solely to identify a human being; health-related data; data concerning a person's sex life or sexual orientation".

5.2.2 The adversary model

Knoxly aims to protect web users from the accidental spread of any inappropriate content, especially private or sensitive informa-

[3] Art. 4 (13), (14), (15) and Art. 9 GDPR. `https://bit.ly/2VJc2kX`

tion about themselves. We mainly consider the risk of improper dissemination towards two audiences: (i) followers or friends who receive updates on the user's posts; (ii) external stalkers (both humans and bots), who peek into a target user's social network posts. Both are likely to know the user's offline identity. *Knoxly* provides primarily privacy awareness, does not prevent the user from publishing (sensitive) content, nor does it prevent the recipient from viewing the content. *Knoxly*'s idea assumes that trackers can explore the OSN through the user interface or collect data using an automated crawler via the reference OSN APIs or scraping. Finally, it does not consider retraction/cancellation of previous posts, nor does it analyze them.

5.3 A ML approach to provide privacy awareness

In this section, we explain the methodology followed in pursuing the objectives presented in Section 5.1. In particular, we aim to lay the foundation of a system capable of: (i) detecting sensitive or personal data in a text, (ii) understanding the semantics of a text written in natural language, (iii) detecting potential privacy risks (i.e., dissemination of sensitive information), (iv) adapting to the user preferences. At the same time, the system should be: (i) lightweight, (ii) fast, (iii) effective.

To this end, loosely based on Grammarly[4], we headed towards (a) regular expression and dictionaries, and (b) ML-based solutions. In particular we conceived a system encompassing several modules (see Figure 5.2): *Keyword module* (Section 5.3.1), *Topic module* (Section 5.3.3), *Sensitiveness module* (Section 5.3.4), *Customized module* (Section 5.3.5).

[4] https://www.grammarly.com

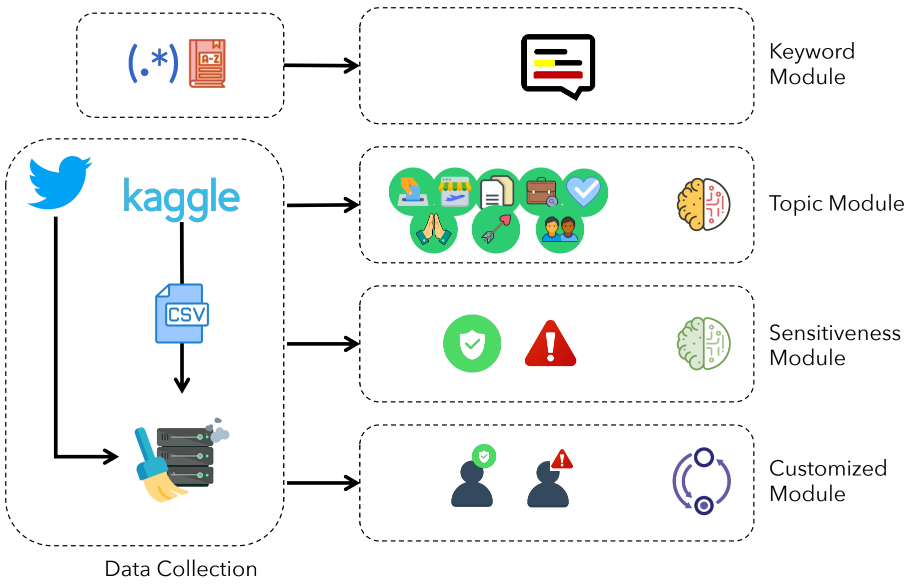


Figure 5.2: Main phases of the methodology adopted: (1) Implementation of regular expressions and identification of suitable dictionaries; *(2) Keyword module*, to create a module capable of highlighting suspicious personal or sensitive data in a text; *(3) Data collection*, identification, collection and pre-processing of data useful for the subsequent phases; *(4) Topic module*, to create a model that given a sentence is able to recognize the topic to which it belongs; *(5) Sensitiveness module*, to create a model that given a sentence is able to understand if it contains potentially personal and sensitive data; *(6) Customized module*, to create a model capable of understanding what the sensitive data are for a user, adapting to his perception of privacy.

5.3.1 Keyword module

This module aims at identifying, when a user is typing a message, whether there are Personally Identifiable Information (PII) data, sensitive data (SD) and quasi-identifier (QI).

To identify such data, we rely on a *lexing* task, that is the usage of regular expressions and dictionaries. In Table 5.1 we deliver details about which data is identified through the former (*RegExp*), and which ones via the latter (*Dict*).

Data	**Description**	**Lexing**
Name and surname	Name and surname most widespread in IT and USA	Dict [355, 356]
Address	Address defined as the main street toponyms (IT and USA), and a consecutive string	RegExp
Email address	-	RegExp
Identifier number	Social Security Number (SSN) (9 numbers) and Codice Fiscale for IT (16 alphanumeric)	RegExp
Passport number	Unique code associated to international passports	RegExp
IP address	IPv4 and IPv6	RegExp
Car license plate	Car license plate for EU countries based on the standard format	RegExp
Driving license code	For EU countries only based on the standard format	RegExp
Bank account	Credit card code, or IBAN code	RegExp
Date of Birth	A date in one of the following formats: dd-mm-yy(yy), yy(yy)-dd-mm, mm-dd-yy(yy), or with three letters months	RegExp
Birth location	IT cities, or EU and USA most populated cities	Dict [357]
Telephone number	-	RegExp
Race or ethnic group	-	RegExp
Religious beliefs	terms indicating religious opinions of beliefs	Dict
Union	Terms defining a Union membership in EU and USA (defined by International Confederation of Free Trade Union)	Dict
Health	Terms indicating disease, based on the well-known ICD09 classification (International Classification of Diseases)[5]	Dict [358]
Sexual orientation	Terms defining information about sexual lifestyle or sexual orientation	RegExp
Zip code	-	RegExp

Table 5.1: Data identifiable with *Keyword module* via regular expression (RegExp) or dictionaries (Dict).

The set of keywords here provided can be considered complete about the "length", that is it covers all the currently known types of PII, SD and QI, but not concerning the "depth", i.e., there are several widely-spoken languages or (less known) keywords not considered. This due to the task's (time-consuming) nature to be accomplished by manually coding every regular expression and/or building dictionaries.

5.3.2 Data collection

This phase consists in collecting the data useful for the following steps. In order to create the dataset on which the models of (a) *Topic module*, (b) *Sensitiveness module* and (c) *Customized module* will be trained, we must first collect the raw data (i.e., not processed). Based on the available literature [359, 21] and Art. 9 of GDPR[6] on sensitive data, seven topics of particular interest have been identified: "politics", "travel", "health", "job", "sexual orientation", "racism" and "religion".

Topic	**Dataset name**	**Reference**	**# entries**
Politics	Election Day Tweets	[360]	393764
Health	Medical Transcriptions	[361]	2348
Health	Medical Speech, Transcription, and Intent	[362]	706
Job	AMAZON Job Skills	[363]	2505
Travel	Twitter US Airline Sentiment	[364]	14427
General	The Movies Dataset	[365]	44306

Table 5.2: Dataset chosen on Kaggle.com for each topic of interest.

Once the topics were defined, suitable raw datasets were identified. In particular, for the topics "health", "politics", "job", and "travel", the raw datasets made available from the Website kaggle.com were downloaded. Table 5.2 shows the datasets selected for each topic with the relative number of elements and the link to download them.

In addition, we decided to introduce a topic that we will call "general" which will be used to increase the heterogeneity of the data, creating noise. It is clear that a ML model based only on topics potentially rich in sensitive and personal data, would classify any text in one of the identified categories: this would generate misclassification because there are texts that do not fall into any of the categories between "health", "politics", "job", "travel", "racism", "religion", and "sexual orientation". In the topic "general" there are some movie plots.

[6] `https://gdpr-info.eu/art-9-gdpr/`

For the topics “sexual orientation”, “racism” and “religion” there were no datasets available. Thus, we gathered information from Twitter data. Specifically, we collected data about the aforementioned topics over the period of 3 months from September 2020 to November 2020. As a result, we have gathered 20,000 tweets. To collect the data, as a general rule, we defined the queries with words found in [21] as search keywords for the Twitter Standard API Search. Concerning the topic “racism” we merged those words with a set of English common *hate-words*, available through *EnglishClub*[7]. In Table 5.3, we depict an example of search queries performed on Twitter and the obtained result. For more information about the employed keywords for each topic of interest, we refer the reader to the Appendices .3.

Topic	**Query**	**Result**
racism	(africa OR african OR african american) (airhead OR annoying OR arsehole)	“*I love my African American father he is litterally perfect but the nigga is annoying on god*”
Religion	Allah OR Jah OR Jehovah OR Jesus OR Messiah OR agnostic OR almighty	“*May Allah call all of us to Madinah Aameen*”
Sexual Orientation	agender OR androgyny OR asexual OR bi-sexual OR butch OR cishet	“*I’m confused. Can someone be a lesbien and asexual? What does that mean?*”

Table 5.3: Examples of search query for Twitter Standard API Search and related result.

Preprocessing We removed the mentions (e.g., @something), urls, and then texts with one only word. We observed that the texts in the “job” raw dataset are particularly long (average length of the texts is 968 characters) and this can represent a compliance

[7]`https://www.englishclub.com/ref/Slang/Insulting/`

problem compared to the other raw datasets considered. To reduce the size of each entry belonging to the "job" category, we divided the job advertisement into various sentences. This was done using the sentence splitter[8], provided for Python based on the work by the researchers Philipp Koehn and Josh Schroeder [366]. For what concerns the topics "sexual orientation", "racism" and "religion", we manually filtered out irrelevant tweets, not belonging to the specific topic, that we collected due to the match with a few identified keywords actually used as synonyms for other concepts. For this task we employed three independent annotators that were experts in privacy. For measuring the agreement among them, we employed the Fleiss' Kappa (adaptation of Cohen's Kappa [367] for 3 or more raters) between them obtaining $k = 0.94$, thus an almost perfect agreement. We obtained the following amount of relevant samples: 2,048 for the topic "racism", 1,696 for the topic "religion", 4,743 for the topic "sexual orientation".

Once the preprocessing phase for the raw dataset was completed, we obtained a total of 2,453,348 samples split among the various topics of interest.

5.3.3 Topic module

We aim at defining and implementing an intelligent module capable of understanding the semantics of a text written in natural language, i.e. which is able to understand what the user is talking about.

To this end, the following methodology has been defined (see Figure 5.3):

- Dataset building: it details (a) the technique(s) used to convert the texts of the dataset into vectors of real numbers (samples) to input into the classifier, (b) the samples labeling strategy;

- Validation: splitting the dataset into training and testing,

[8]`https://github.com/berkmancenter/mediacloud-sentence-splitter`

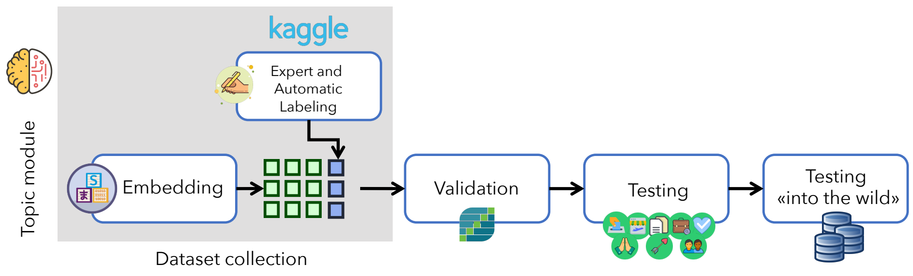


Figure 5.3: Main phases for the creation of the Topic module: (*Embedding*) the sentences of the various datasets are transformed into fixed-length vectors; (*Automatic Labeling*) based on the dataset of origin, each sample (embedding) is automatically labeled; (*Validation*) through a k-fold cross-validation, the classifier based on supervised learning is validated and the hyper-parameters are tuned using the training set; (*Testing*) once the best parameters for the classifier have been obtained, testing is carried out on the test set; (*Testing "into the wild"*) further testing is performed on new data.

then validation of well-known ML methods using the k-fold cross-validation on the training set;

- Testing: evaluating the performance of the trained ML methods on the test set;
- Testing "into the wild": further evaluation of the performance of the best ML method found in previous stages, on a much larger dataset.

Dataset building

For building this module, we decided to use a subset of the entire dataset, comprising 1000 samples for each topic, selected by a `random` function with Python. Hereinafter we will refer to this dataset as *ds1000*. We then have tackled the phase of the text representation and labeling.

Text representation The related works (see Section 5.6) used k-grams of word embeddings which are known to be computation demanding and thus, computationally unfeasible with respect to our objectives, i.e., a (usable) tool providing awareness in real-time for end users. Therefore, based on previous experiences [368, 369, 370, 371, 372], Google mUSE was exploited to represent the texts (see Section .2 for more details), because we argue that the sentences belonging to the same topic are similar to each other and consequently also the vectors resulting from the embedding computation will be similar. Each text is then embedded into an array of 512 elements.

Labeling After effectively representing the dataset of texts written in natural language, we moved on to the labeling phase, which consists in assigning a class to each sample. Concerning the datasets found on Kaggle (topics politics, health, job, travel, and general), the task is automatic because the samples are already implicitly labeled, based on the raw dataset they come from. With regard to the datasets downloaded from Twitter (topics racism, religion, and sexual orientation), we already performed this task while filtering out irrelevant tweets from our datasets (see Section 5.3.2); therefore, selected tweets are already labeled.

In Table 5.4 the labeling strategy is shown together with the main information about the resulting dataset for the *Topic module*.

Validation

We first split the dataset (Table 5.4) into an 80% reserved for training and a 20% reserved for testing through the function *train_test_split* from scikit-learn[9] library (with a *stratified* approach).

The validation phase was carried out using the training set. In particular, k-fold cross-validation was used, a resampling procedure used to evaluate ML models on a limited sample of data. The procedure has as its only parameter k which represents the

[9]https://scikit-learn.org/stable/

Topic	**# sample**	**Label**
Politics	1000	0
Health	1000	1
Job	1000	2
Travel	1000	3
General	1000	4
racism	1000	5
Religion	1000	6
Sexual Orientation	1000	7

Table 5.4: Labeling strategy and main information on the dataset resulting from the *Data collection* phase.

number of subsets that must be created starting from the original sample, in our case $k = 10$.

This phase was also exploited for hyper-parameter tuning using the *GridSearchCV* method from scikit-learn. In this phase, we aim at maximizing the (weighted) F-score value (see Section **??**), appropriately setting the `scoring` parameter of *GridSearchCV*.

We compared the performance obtained by the most used ML methods available in literature: *Random Forest* (RF), *Support-vector machines* (SVM), *Adaptive Boosting* (Ada), *Multilayer perceptron* (MLP). Specifically, for each ML method, we analyze the following hyper-parameters (shown in *italic*):

- RF – Its performance mainly relies on *n_estimators*: it has been tested from `100` a `1000` estimators. The best result has been obtained in the range [`300`, `550`].

- SVM – It has been tested on different *kernel* (polynomial, sigmoid, radial) and optimized with respect to the *penalty parameter C* (from `0.001` to `1000`). The best results have been obtained with `radial` kernel ed C in the range [`10` e `100`].

- KNN – The performance have been evaluated through *n_neighbors* parameter in a range from `3` to `10`, and *weights* set as *uniform* or *distance*. Best results are with *n_neighbors* between 6 and 8, and using *distance* weight.

- Ada – The performance of this model are based on *n_estimators*: we tested from 100 to 1000 estimators, and we get that the optimal number is between 750 e 1000.

- MLP – The model is primary based on the *hidden layers size*. This parameter's dimension has been tested in the range between 100 and 450. The best *hidden layers size* results have been found between $\frac{3}{5}$ e $\frac{4}{5}$ of the input layer size. Furthermore, we adopted *lbfgs optimizer*, which is proved to converge faster and showing better results on (relatively) small dataset [373].

The results of this phase are available in Table 5.5.

Testing

Once obtained the best parameter for our ML methods we tested their performance on the test set, i.e. the 20% of the unused *ds1000*. Table 5.5 shows the results of this phase. We add here a method named "Rand. pred." that is a random baseline for assessing the effectiveness our approach.

ML method	**Validation**	**Testing**							
	(avg. F1 +/- std. err.)	**P**	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
RF	0.97 (+/- 0.01)	0.96	0.99	0.98	0.97	0.95	0.92	0.95	0.94
SVM	0.99 (+/- 0.01)	0.99	1.00	0.99	0.99	1.00	0.98	0.98	0.99
KNN	0.98 (+/- 0.01)	0.98	1.00	0.99	0.99	0.99	0.96	0.96	0.98
Ada	0.96 (+/- 0.01)	0.96	0.99	0.98	0.98	0.96	0.93	0.94	0.95
MLP	0.98 (+/- 0.01)	0.97	0.99	0.98	0.99	0.97	0.95	0.96	0.98
Rand. pred.	//	*0.13*							

Table 5.5: F1-score results in both the *validation* and *testing* phases for all the ML methods compared on the ds1000 dataset. **Legenda:** P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = racism, Rel = Religion, SO = Sexual Orientation.

We remark that the goal of this step is to develop a method to classify the topic of a text written in natural language with reference to the four most discussed topics in the privacy literature, plus a "general" topic. In particular, we want to have a classifier which obtains good performance on all the classes both in precision and recall, hence the comparison made on the F1-score which includes both metrics.

The performances of the compared models are very similar to each others (an all are definitely better than the *Rand. pred.* method). To evaluate the statistical significance of the results, the Shapiro-Wilk [374] statistical test was first used to evaluate the distribution of the data. The observed distribution is not a normal distribution, so a non-parametric test was used, in particular the Kurskal-Wallis [375] with $p - value = 0.05$. This test resulted in a significant statistical difference between the methods ($H = 14.44$, $p < 0.05$), where the SVM shows better performance than the other methods.

Thus, the *Topic module* is based on the SVM classifier. Such a trained model has been dumped into a `pickle` object[10]. Hereinafter we will refer to *Topic classifier* as theSVM classifier embedded/underlying in the *Topic module*. For more information about the obtained results see Appendix **??**.

Testing "into the wild"

Topic classifier has undergone further testing. We decided to measure the classifier's performance on a much larger dataset, named *wild dataset*. Wild dataset is made up of all others sentences available in our raw dataset (Section 5.3.2), with samples represented as shown in Section 5.3.3. Thus it comprises politics, health, job, travel, general, racism, religion, and sexual orientation sentences not belonging to *ds1000* previously seen.

Table 5.6 shows the performance obtained by *Topic classifier* within the "into the wild" experiment.

[10] `https://docs.python.org/3/library/pickle.html`

ML method	**Metric**	**Topics** (Testing "into the wild")							
		P	H	J	T	G	Rac	Rel	SO
SVM	Accuracy	0.99							
	Precision	0.99	1.00	0.99	0.98	0.99	0.98	0.97	0.98
	Recall	0.97	1.00	0.99	1.00	0.99	0.98	0.99	0.97
	F1-score	0.98	1.00	0.99	0.99	1.00	0.98	0.98	0.98

Table 5.6: Performance of *Topic classifier* (based on SVM) on the *wild dataset*. P = politics, H = health, J = job, T = travel, G = general, Rac = racism, Rel = religion, SO = sexual orientation.

We observe that *Topic classifer* exhibit performance on par with those seen in the testing phase (Section 5.3.3).

5.3.4 Sensitiveness module

With this module we aim at understanding whether a sentence written in natural language contains or not personal and sensitive information that may intrude the privacy of the user which is disseminating that content. An easy example of sentences' sensitiveness is sketched in Table 5.7.

To analyze the sensitivity level of a sentence we have identified two possible approaches shown in the Figure 5.4:

1. creating a single classifier that is trained to recognize and identify whether a sentence contains sensitive/personal data or not; in this case the embedding is the input of the *sensitiveness classifier*;

2. creating five different classifiers, one for each topic, capable of recognizing and identifying whether a phrase that talks about a certain topic (i.e., that belongs to one of the topics referred to in Section 5.3.3) contains sensitive/personal data or not; in this case the embedding is the input of a *topic-specific sensitiveness classifier* chosen on the topic classification basis.

Sentence	**Topic**	**Sensitive**
Last night I have not slept	Health	✗
I suffer from insomnia	Health	✓
New York is a beautiful city	Travel	✗
I'm landing to new York right now	Travel	✓
A computer scientist needs several skills and soft skills	Job	✗
Bob works as a computer scientist for BigCompany	Job	✓
I'm not sure if I have any left leaning friends in North Carolina but if I do please get to the polls for voting	Politics	✗
Please America, vote for Donald Trump!	Politics	✓
Why is every mf on twitter bi. I know its double the chances but still god damn	Sexual Orientation	✗
Annette is bi though she may only be into Mercedes as far as women go	Sexual Orientation	✓
Don't be fucking racist to Asian people because you think they have coronavirus!! It's that simple!!!!	racism	✗
hate you black bastards you stink! i hate ya black skin.. i hate ur black pants..i hate black pepper!	racism	✓
Haven't the Jewish people suffered enough?	Religion	✗
Glory to GOD May He protect love comfort humanity in Jesus name Grace - forgiveness- healings	Religion	✓

Table 5.7: Example of contents sensitive (✓) and not sensitive (✗).

In both cases the methodology sketched in Figure 5.5 has been defined, following the footsteps of the one previously seen (Section 5.3).

Data collection

In order to create this module, we need a dataset of sensitive and non sensitive sentences. Unfortunately this kind of dataset is not available elsewhere, thus we built it from scratch. We exploited a subset of *ds1000*, in particular 200 samples for each topic to

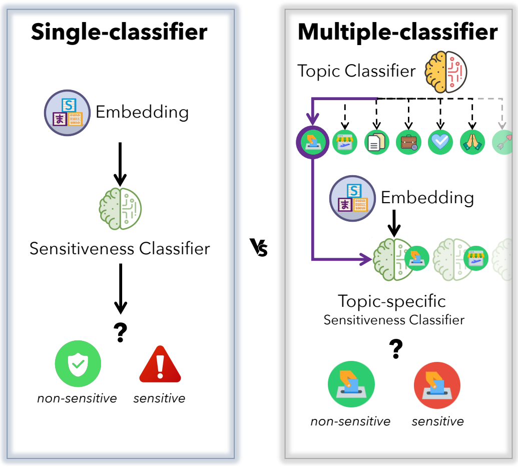


Figure 5.4: The two approaches to analyze sensitiveness of a sentence. Single-classifier vs. Multiple-classifier

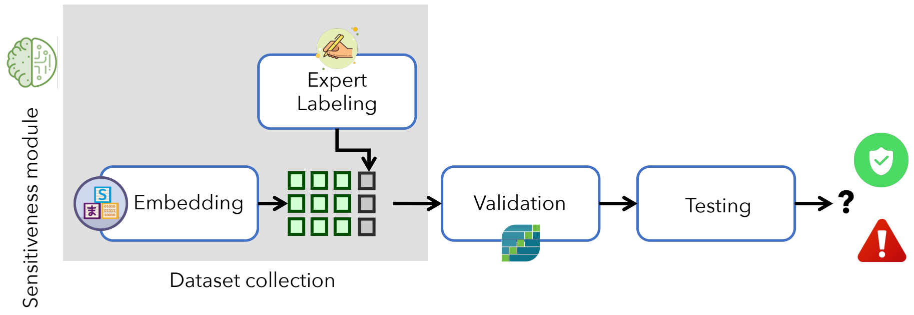
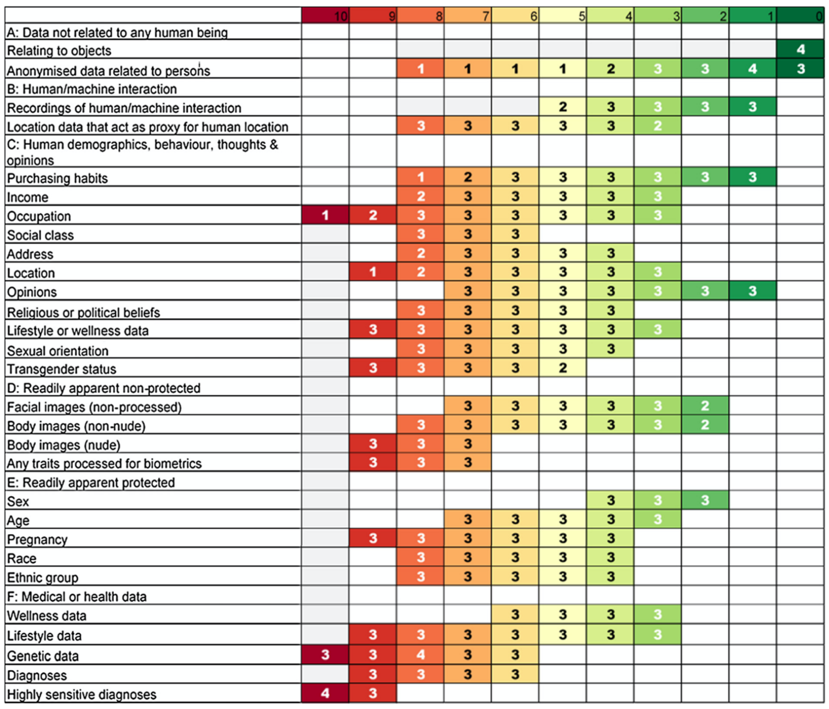


Figure 5.5: Main phases to create the *Sensitiveness module*: (*Embedding*) the sentences are transformed into fixed-length vectors; (*Expert labeling*) through a series of well-defined rules [376], a domain expert manually labels the samples; (*Validation*) through a k-fold cross-validation, Random Forest classifier is validated, and the hyper-parameters are tuned, using the training set; (*Testing*) once the best parameters for the classifier have been obtained, testing is carried out on the test set;

tackle this project phase. Hereinafter we will refer to this dataset as *ds200*. Samples in such a dataset have been represented as previously seen with mUSE.

The labeling in this case has been performed manually. The labeling strategy has been defined based on the guidelines provided by Rumbold et al. [376], that have worked on categorizing a sensitivity data spectrum. To ease the understanding of our strategy we report the outcome of [376] in Figure 5.6.



	10	9	8	7	6	5	4	3	2	1	0
A: Data not related to any human being											
Relating to objects											4
Anonymised data related to persons			1	1	1	1	2	3	3	4	3
B: Human/machine interaction											
Recordings of human/machine interaction						2	3	3	3	3	
Location data that act as proxy for human location			3	3	3	3	3	2			
C: Human demographics, behaviour, thoughts & opinions											
Purchasing habits			1	2	3	3	3	3	3	3	
Income			2	3	3	3	3	3			
Occupation	1	2	3	3	3	3	3	3			
Social class			3	3	3						
Address			2	3	3	3	3				
Location		1	2	3	3	3	3	3			
Opinions				3	3	3	3	3	3	3	
Religious or political beliefs			3	3	3	3	3				
Lifestyle or wellness data		3	3	3	3	3	3	3			
Sexual orientation			3	3	3	3	3				
Transgender status		3	3	3	3	2					
D: Readily apparent non-protected											
Facial images (non-processed)				3	3	3	3	3	2		
Body images (non-nude)			3	3	3	3	3	3	2		
Body images (nude)		3	3	3							
Any traits processed for biometrics		3	3	3							
E: Readily apparent protected											
Sex							3	3	3		
Age				3	3	3	3	3			
Pregnancy		3	3	3	3	3	3				
Race			3	3	3	3	3				
Ethnic group			3	3	3	3	3				
F: Medical or health data											
Wellness data					3	3	3	3			
Lifestyle data		3	3	3	3	3	3	3			
Genetic data	3	3	4	3	3						
Diagnoses		3	3	3	3						
Highly sensitive diagnoses	4	3									

Figure 5.6: The data sensitivity spectrum sheet developed in [376].

The lines of the spectrum contain the category which a sentence can belong to, and how sensitive it can be. The *sensitivity level* varies from 0 (non sensitive) to 10 (very sensitive). The numbers in the cells ranges between 0 and 4: they represent the *likelihood* that the topic described in the row has the sensitivity level described in the column. For example, let's take the topic *Relating to object*: it has a sensitivity level of 0 and the cell shows the number 4; this means that when we talk about topics related to objects they will certainly be non sensitive. Based on the above, we defined a set of rules to label a sentence as sensitive (✓) or non sensitive (✗). The criteria defined in Table 5.8 resulted from the application of the following method.

Let $freq$ be the likelihood values used by [376]. Let $freq[i]$ be the likelihood value for the sensitivity level i, where $i \in [0..10]$. Levels $i < 5$ indicate a low sensitivity level (low risk), instead levels $i >= 5$ indicate a high sensitivity level (high risk). We then compute the total sensitiveness score as $\sum_{i=x}^{N} freq[i]$, $N = \{4, 10\}$, $x = \{0, 5\}$. Thus, we label a sample as non sensitive if:

$$\sum_{i=0}^{4} freq[i] > \sum_{i=5}^{10} freq[i] \tag{5.1}$$

Vice versa, if sensitive.

Rule	**Label**
Data relating to objects	✗
Anonymized data relating to persons	✗
Data relating to human-machine interaction	✗
Data relating to human location	✓
Data relating to shopping habits	✗
Data related to wage	✓
Data relating to job	✓
Data relating to social class	✓
Address and location	✓
Clear political or religious opinion	✓
Other opinions	✗
Lifestyle data	✓
Sexual orientation	✓
Sex in general	✗
Data relating to pregnancy	✓
Data relating to race and ethnic group	✓
Health data	✓

Table 5.8: Rules used to label *ds200*. ✓indicates that a sentence including those data is sensitive, ✗vice versa.

We defined such a strategy, which turns the problem into a binary classification task[11], because, as formulated by [376], the problem would have fallen in the regression field, a very hard task for ML models. Regression requires large scale, annotated, datasets. This is clearly possible to obtain by employing a quite large number of annotators but (1) in this work it is not our main objective, (2) we need expert annotators for this task. Therefore, we employ three expert in the privacy field as annotators. We have computed the inter annotator agreement through the Fleiss' Kappa, resulting in $k \in [0.84, 0.90]$, that is an almost perfect agreement for each topic, except "religion". In this case, we obtained a $k = 0.77$, i.e., a "substantial agreement". Once labeled samples in *ds200* we obtained the dataset whose information are summarized in Table 5.9.

Topic	**# sample**	**# sensitive**	**# non sensitive**
Politics	200	100	100
Health	200	100	100
Job	200	80	120
Travel	200	100	100
General	200	97	103
racism	200	114	86
Religion	200	114	86
Sexual Orientation	200	89	111
Total	1600	797	803

Table 5.9: Dataset *ds200* used for the *Sensitiveness module*.

Validation

The dataset for both approaches have been split into 80% for training and 20% for testing with *train_test_split* (stratified). The training has undergone the validation phase. We performed a k-fold cross-validation with $k = 10$ using *GridSearchCV* and searching for the optimal accuracy score since datasets are well balanced. In

[11]In the implementation, we used 1 for labeling sensitive data and 0 for the non-sensitive ones.

Table 5.10 we show the best results obtained in this phase with both approaches, i.e., single-classifier and multiple-classifier (see Figure 5.4), by adopting the same approach previously seen in the Topic module validation (Section 5.3.3).

Single-classifier With this approach, a single classifier is able to detect sentences containing sensitive and/or personal data. The advantage is that of having a single sensitiveness classifier capable of identifying sensitive content in a "context-free" fashion; the disadvantage is the fact that this type of classifier has a coarse-grained knowledge thus it can provide misleading classification to the final user.

Multiple-classifier With this approach, we create multiple classifiers, one for each topic of interest, that is eight. Such classifiers are capable to detect sensitive content based on the specific topic the sentence is about. The advantage of this approach is that we obtain more fine-grained knowledge; the disadvantage is having more classifiers for the same task.

In the following, we summarize the best configurations obtained in this phase for each ML method compared:

- RF – The number of *estimators* parameter is optimal in the range [`500 1000`].
- SVM – Best results have been obtained with `radial` *kernel*, and the C parameter between `1` and `10`.
- KNN – We obtained the optimal *n_neighbors* in the range `4` to `5`, and `uniform` as optimal for *weight* parameter.
- Ada – The optimal number of *estimators* is between `250` e `500`.
- MLP – We used the `lbfgs` optimizer, and *hidden layer size* resulted optimal in the range [`300,350`].

Topic	*Validation (avg. accuracy. ± std. err.)* RF	SVM	KNN	Ada	MLP
P	0.88 ± 0.06	0.82 ± 0.06	0.84 ± 0.08	0.84 ± 0.11	0.82 ± 0.06
H	0.96 ± 0.07	0.89 ± 0.09	0.88 ± 0.08	0.87 ± 0.09	0.91 ± 0.07
J	0.91 ± 0.09	0.90 ± 0.10	0.87 ± 0.09	0.87 ± 0.10	0.89 ± 0.08
T	0.91 ± 0.08	0.83 ± 0.08	0.85 ± 0.08	0.81 ± 0.07	0.86 ± 0.06
G	0.72 ± 0.09	0.69 ± 0.11	0.65 ± 0.15	0.67 ± 0.10	0.72 ± 0.12
Rac	0.88 ± 0.12	0.78 ± 0.10	0.73 ± 0.09	0.65 ± 0.10	0.80 ± 0.10
Rel	0.70 ± 0.12	0.68 ± 0.11	0.71 ± 0.12	0.65 ± 0.12	0.68 ± 0.13
SO	0.77 ± 0.14	0.73 ± 0.10	0.70 ± 0.12	0.72 ± 0.14	0.73 ± 0.14
All	0.83 ± 0.05	0.79 ± 0.07	0.78 ± 0.07	0.81 ± 0.08	0.81 ± 0.05

Table 5.10: Accuracy results obtained in the Validation phase for each ML method compared on both the task (Single-classifier, i.e., one only for all the topics, and Multiple-classifier, i.e., a specific classifier for each topic). P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = racism, Rel = Religion, SO = Sexual Orientation, All = all topics.

Testing

Here we report the results obtained with both approaches for the testing phase, performed on the test set, i.e., the 20% of *ds200* unseen by classifiers. As we can see in Table 5.11, the single-classifier approach is worse than the multiple-classifier approach, where the cases of "general", "religion", and "sexual orientation" topics show performance slightly lower than the single-classifier setting. Concerning the *general* topic, this happens because of the heterogeneity of sensitive contents within. If for *politics* we have only sensitive contents regarding the political beliefs disclosed, for *health* we have only sensitive contents regarding data related on pregnancy and health, for *travel* we have only sensitive contents including data concerning human location, and for *job* we have only sensitive contents including data related to job and/or wage, for the *general* topic we can have sensitive contents about for example lifestyle data which are a broader category. Therefore getting the sensitiveness of a text that falls into *general* topic is a more challenging task than for the other topics considered in this work. The same applies for the case of "religion" where also the expert annotators did not get a perfect agreement. In this topic, several citations taken from holy books fall. Thus, it is hard to understand if the citation is made to disclose the user's religious belief or not.

ML method	*Testing (accuracy)*								
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**	**All**
RF	0.77	0.88	0.85	0.85	0.72	0.80	0.71	0.75	0.80
SVM	0.83	0.88	0.85	0.75	0.62	0.73	0.75	0.78	0.77
KNN	0.88	0.88	0.75	0.80	0.75	0.78	0.65	0.76	0.77
Ada	0.83	0.90	0.90	0.75	0.58	0.78	0.70	0.75	0.78
MLP	0.85	0.90	0.90	0.80	0.62	0.75	0.70	0.75	0.79
Always Sens.	*0.50*	*0.51*	*0.40*	*0.50*	*0.56*	*0.57*	*0.57*	*0.44*	*0.50*

Table 5.11: Accuracy results obtained during the Testing phase for each ML method compared. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = racism, Rel = Religion, SO = Sexual Orientation, All = all topics.

For this reason we have developed the *Sensitiveness module* with the *multiple-classifier* approach. To evaluate the effectiveness of the different ML methods, we first employ the Shapiro-Wilk test [374] to check if the normal distribution model fits the observations. The test fails, thus we apply the non-parametric test Kruskal-Wallis [375] with $p - value = 0.05$, obtaining no significant statistical difference between the ML methods ($H = 0.15$, $p < 0.05$). Therefore, given the wide-usage in real-time applications (e.g., see [377]) this module is based on RF classifiers, which have been dumped as `pickle` objects. For a closer look at RF performance, see Table 5.12.

5.3.5 Customized module

Privacy is a subjective perception, in which each user has his own level of confidentiality concerning his private information [378].

		Validation **Accuracy ± std.**	*Testing* **Accuracy**	**Precision (ns/s)**	**Recall (ns/s)**	**roc_auc**
RF	**P**	0.88 ± 0.06	0.77	0.70/0.76	0.80/0.65	0.78
	H	0.96 ± 0.07	0.88	0.89/0.86	0.85/0.90	0.94
	J	0.91 ± 0.09	0.85	0.81/0.92	0.96/0.71	0.93
	T	0.91 ± 0.08	0.85	0.80/0.94	0.95/0.75	0.97
	G	0.72 ± 0.09	0.72	0.70/0.73	0.82/0.60	0.74
	Rac	0.88 ± 0.12	0.80	0.75/0.78	0.80/0.85	0.81
	Rel	0.70 ± 0.12	0.71	0.67/0.70	0.75/0.64	0.72
	SO	0.77 ± 0.14	0.75	0.70/0.74	0.78/0.65	0.76

Table 5.12: Results obtained developing by RF in the *Sensitiveness module*, both for validation and testing phases. *ns* = non sensitive, *s* = sensitive, *std.* = standard error.

More importantly, the attitude towards privacy varies from a topic to another. It follows that the implementation of a system that does not take into account the needs and preferences of users can result in an obstructive, annoying, or even useless tool. Therefore, we need to develop a system that somehow adapts to users' needs and preferences.

There are different techniques suitable for this kind of task, such as reinforcement learning (e.g., [379]), Q-learning (e.g., [380]), agents based on LSTM Network (e.g., [381]), and so on. These models learn based on feedback received from the environment in which they are running: for example, user feedback is useful for training recommendation systems [382].

To learn users' attitudes towards personal privacy, we defined and implemented an online learning model which, based on the user's perception of sensitivity, will alert the user only for the contents that are sensitive for him/her. To this end, we drew inspiration from a priority inbox system [383, 384]. Such a system can understand which emails, according to a specific user, have a higher priority than the others. To achieve this goal, the authors relied on the interactions that users had with incoming emails: for example, if an email was opened after more than seven days, then it was not an important email; thus, a low priority should be assigned to it.

Making an appropriate parallelism, this scenario is quite similar to our case, in which the end-user, interacting with the online model, should provide feedback regarding his personal perception towards privacy. The *Customized module*'s objective is to have a ML-based client-side model capable of learning the user's preferences. Implementing this module on the client-side is mandatory since (a) perception towards privacy is private information, and (b) a privacy prevention tool does not have to violate privacy to prevent it. For this reason, a server-side classification that would involve privacy leakage (towards first-part) cannot be carried out. This constraint leads us to the developing of a model that is *light* and *simple*. *Light* in such a way that it does not slow down the user's normal web browsing; *simple* to be implemented on the client-side.

In the light of above, our choice is the Passive-Aggressive online learning method, first presented in [385]. In particular, we adopted the *PAII* method.

Let D be a dataset.

$$D = \begin{cases} X = \{x_1, x_2, \dots, x_t, \dots\}, x_i \in \mathbb{R}^n \\ Y = \{y_1, y_2, \dots, y_t, \dots\}, y_i \in \{-1, +1\} \end{cases}$$

The t index has been chosen to mark the temporal dimension of the problem. In this case, in fact, the samples can continue to arrive indefinitely. Of course, if they are drawn from the same data-generating distribution, the algorithm will continue to learn (without major parameter changes), but if they are extracted from a completely different distribution, the weights will be slowly "forgotten" in favor of the new distribution. In our case, we have a binary classification problem (-1 non-sensitive data, 1 sensitive data).

Given a vector w, the prediction is obtained as:

$$\tilde{y}_t = sign(w^T \cdot x_t)$$

Such a algorithm is based on the Hinge-loss function (the same used by Support Vector Machines) [65]. The algorithm's update rule is:

$$w_{t+1} = w_t + \frac{max(0, 1 - y_t(w^T \cdot x_t))}{\|x_t\|^2 + \frac{1}{2C}}$$

where C is the aggressiveness parameter (the higher the C value the more aggressive is the classifier, resulting in a higher risk of destabilization in presence of noise). The vector w is updated with a factor whose sign is determined by $y(t)$ and whose size is proportional to the error. Observe that if there is no wrong classification the numerator becomes 0, then $w(t+1) = w(t)$. For more details, we refer the reader to [385].

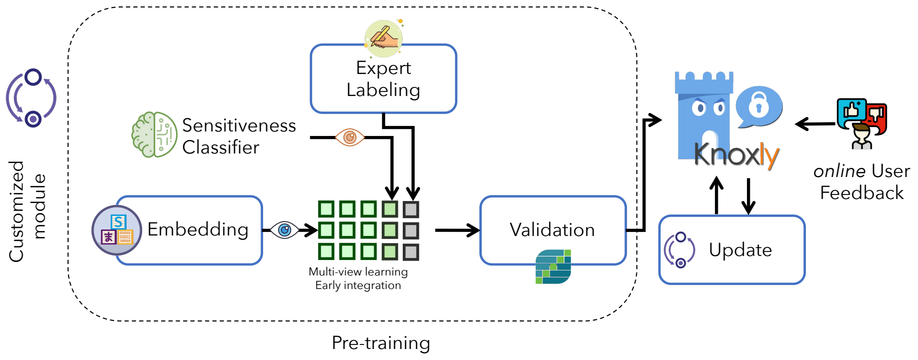


Figure 5.7: Main steps followed for the implementation of *Customized module*. Left part is for a pre-training phase in order to provide users with a pre-trained model.

Our idea is to provide the user with an already pre-trained system that slowly adapts to his needs, overcoming a well-known issue faced in many research projects, i.e., the cold starts problems (see, for instance, [386]). For this purpose we must adopt a transfer-learning / multi-view learning [387, 388, 389] strategy (see Figure 5.7) to train our model.

We want to train the classifier within the *Customized module* with a multi-view that includes both the embed (single-view) and the evaluation of the sensitiveness classifier in the *Sensitiveness* *module* (single-view). We adopted the *early integration* technique in particular. The early integration consists in concatenating the single views that are the features associated with the embed and the results in probability obtained via *Sensitiveness module*; in this way, each combination (concatenation of two or more single views) represents a sample in the data sets. Therefore, we obtain vectors of 514 elements (512 + 2) where the first 512 elements are the embed of the sentence and the last 2 are the probability associated with non-sensitive and sensitive labels, respectively.

Validation

We exploited the dataset *ds200* integrated with the classification made by the matching Topic-specific sensitiveness classifier in the *Sensitiveness module*. For instance, the samples related to the topic "health" have been integrated with the class probabilities returned by the Health-sensitiveness classifier; the same for all other cases.

The datasets have been split, with a stratified approach, into 80% for training and 20% for testing using *train_test_split* function. Subsequently, we validated the models on the training set, via 10-fold cross-validation with the *GridSearchCV* method, in which we tried to optimize the aggressiveness parameter C. However, it should be noted that this parameter is actually very user-based, thus the best method for its estimation would be by carrying out a user study: it is possible that they prefer a very aggressive classifier or, on the contrary, they are more satisfied with a system that learns slower. We tested the following C values: `0.001`, `0.01`, `0.1`, `1`, `10`, `100`. The best parameter was found to be for all cases, $C = 0.01$.

Testing

We also performed a testing phase on the test set. Results of this phase can be found in Table 5.13. We can observe that the Customized module exhibits better performance compared to the sensitiveness module thanks to the multi-view approach. We emphasize this test clarify the effectiveness of the method, more than the accuracy of the module which should be measured concerning how the online learning models cope with the users preferences.

		Validation	*Testing*		
		Avg. Acc. ± std.err	**Acc.**	**Precision (ns/s)**	**Recall (ns/s)**
PAII	**P**	0.87 ± 0.07	0.86	0.81/0.94	0.84/0.89
	H	0.97 ± 0.05	0.98	0.97/0.98	0.96/0.99
	J	0.92 ± 0.08	0.95	0.95/0.94	0.96/0.93
	T	0.93 ± 0.06	0.97	0.97/0.99	0.96/0.95
	G	0.94 ± 0.05	0.96	0.94/0.97	0.96/0.98
	Rac	0.94 ± 0.09	0.95	0.95/0.94	0.96/0.93
	Rel	0.90 ± 0.10	0.90	0.90/0.91	0.91/0.90
	SO	0.98 ± 0.09	0.98	0.97/0.99	0.97/0.99

Table 5.13: Results obtained in the validation and testing phase, for the Customized topic-specific sensitiveness classifiers in the *Customized module*. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = racism, Rel = Religion, SO = Sexual Orientation, *PAII* = Passive-Aggressive classifier. *ns* = non sensitive, *s* = sensitive, *std.* = standard error. *Acc.* = Accuracy

5.4 *Knoxly*, a ML-based Chrome extension to safeguard users privacy

In this section, we describe *Knoxly*, a prototype of a Google Chrome extension that includes the *Keyword module*, *Topic module*, *Sensitiveness module*, and *Customized module* to provide users with privacy awareness towards the information they are about to disseminate on the Web. To the best of our knowledge, *knoxly* is a first prototype of a Google Chrome extension for protecting privacy on the Web which shifts the privacy paradigm to "*privacy as having the ability to control the dissemination of sensitive information*". In particular, we dwell on the workflow (Section 6.2), frontend (Section 5.4.2, and backend (Section 5.4.3) of the tool.

5.4.1 *Knoxly*: workflow

This section presents the *Knoxly* workflow, detailing the various steps that a text written in natural language tackles. Figure 5.8 shows an illustration of the tool's workflow and how the various components interact with each other.

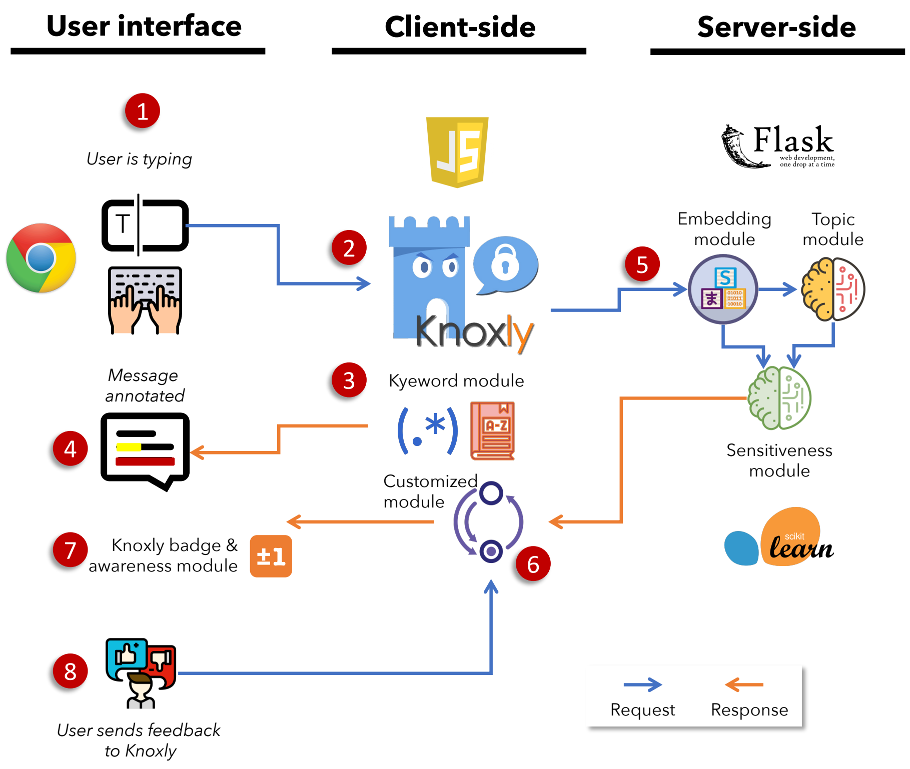
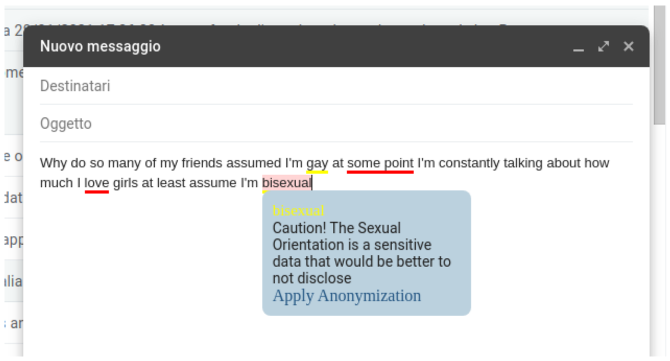


Figure 5.8: *Knoxly* management workflow summarized.

Steps are described in the following:

1. User types text t;
2. The text t is sent to the *Knoxly* plug-in;
3. t is first analyzed with the *Keyword module* using regular expressions and dictionaries (coded in *Knoxly*);

4. The keywords in t that match with regular expressions or dictionaries are highlighted (see Figure 5.9) with the help of a tag `<span>` which is associated with a suitable class CSS (*yellow* for quasi-identifiers [354] and sensitive data, *red* for personal data). It also provides suitable (awareness) tooltips *on hover* the keywords with the possibility to anonymize the keywords reported using the k-anonymity [390] (here the lexical analysis ends);



Why do so many of my friends assumed I'm gay at some point I'm constantly talking about how much I love girls at least assume I'm bisexual

bisexual
Caution! The Sexual Orientation is a sensitive data that would be better to not disclose

Figure 5.9: The Keyword module is devoted to highlight PII, SD, and QI written in the user's message. Here an example on gmail.com.

5. *Knoxly* sends t to the Flask server[12] which embeds the sentence through the embedding module (based on mUSE). Then it sends the embedded t to the *Topic module* and the *Sensitiveness module*. The Topic module, based on the Topic classifier, determines which topic t belongs to. Based on this, the appropriate *Topic-specific Sensitiveness Classifier* in the *Sensitiveness module* is called. The latter returns a vector of two elements (negative sensitivity and positive sensitivity), percentage of the classes, which, together with the embed, will be returned to the client-side.

6. The server sends to the client a vector of 514 elements, the

[12] `https://flask.palletsprojects.com/en/1.1.x/`

embed (512 elements) and the vector of the classes probabilities (2 elements) returned by the *Sensitiveness module*. This response is processed by the *Customized module*, which (a) routes it to the *Customized Sensitiveness classifier* suitable for that topic, (b) evaluates whether the content is sensitive according to the user's perception of privacy.

7. If the *Customized Sensitiveness classifier* classifies the content returned by the server as sensitive, i.e. content that represents a threat to the user's privacy, then it alerts the user by increasing a numeric counter placed in the *Knoxly* badge in Google Chrome, in addition to adding it in a list box - designed to provide awareness - which can be browsed by opening the tool popup.

8. The user has the option to click on "feedback" buttons indicating *Knoxly* (see Figure 5.10): *Ignore* – avoid reporting (as sensitive) that type of text in the future; *Good Job!* – continue to report (as sensitive) that type of text in the future. This operation updates the weights of the reference *Customized Sensitiveness Classifier* for the texts belonging to that topic.

Input heuristics One of the problems faced in the development phase is understanding when it is more appropriate to call *Knoxly*'s AI-based modules. It is clear that the massive usage of backend modules would result in a heavier service, making it slow, annoying and unusable. Furthermore, this would involve the analysis of texts that do not represent real sentences, that is, composed of at least *subject*, *verb* and *complement*, or in any case which include such a number of words that it is possible to understand their semantics. In other words, the question is *"when is the right time to send a sentence to the server and when should we just rely on the Keyword module?"*.

To answer this question we have to define a *input heuristic* that must guarantee the sending of sentences to the *Knoxly* server without the risk of creating a bottleneck.

The input heuristic was defined as follows: let *txt* the vector containing the characters the user typed, n the length of the vector txt, $txt[last]$ the last character in txt and $T = \{".", ";", "!", "?"\}$ the set of characters that are (very often) placed at the end of a sentence written in natural language, then

$$n >= 70 \;\; \& \;\; txt[last] = x \in T \implies \text{send } txt \text{ to the server}$$

. We opt for 70 as threshold to send the messages to the server so that we do not analyze incomplete or not well-structured messages composed on a few words. Some authors tried to overcome this kind of problem by aggregating old tweets of the user [391], but we cannot assume the previous posts concern the same topic of that the user is about to disseminate

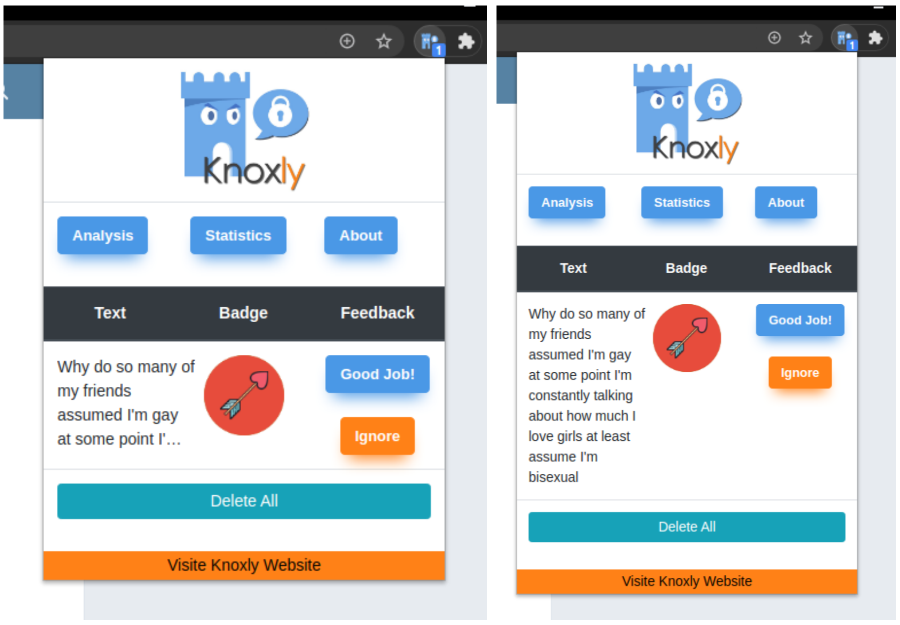
5.4.2 *Knoxly*: frontend

This section details the features of the *Knoxly* frontend. In particular, we will describe the User Interface, and the Keyword and Customized module which have been coded on the client side.

User Interface *Knoxly* User Interface (UI) has been designed and implemented to provide privacy awareness in a natural and intuitive way. *Knoxly* highlights sensitive and / or personal content present in a text through the use of regular expressions and dictionaries. We use *yellow* color for the quasi-identifiers [354] and sensitive data, *red* color for the personal data (Figure 5.9).

By clicking on the icon next to the address bar on Google Chrome, the *Knoxly* popup is displayed (Figure 5.10). Next to the icon, it is also displayed a badge with a numerical counter. This counter is used to provide the user with the number of written sentences/texts classified as threats for his privacy, i.e. sensitive.

The popup UI is mainly composed of three sections/pages: (i) *Analysis*: page dedicated to tool reports. The page also acts as a landing page (Figure 5.10); (ii) *Statistics*: page dedicated to measurements, useful to make the user aware of what he writes about and how much (Figure **??**); (iii) *About*: page dedicated to



Why do so many of my friends assumed I'm gay at some point I'...

Why do so many of my friends assumed I'm gay at some point I'm constantly talking about how much I love girls at least assume I'm bisexual

Figure 5.10: User Interface: *Analysis* section. The user can click on the sentence to show it entirely.

instructions on how to effectively use *Knoxly* (Figure ??). The features of such sections are summarized as follows:

- *Analysis* is made up of a table with three columns. It shows the sentences that *Knoxly* considers sensitive, indicating an awareness icon referring to the topic which the sentence belongs to. The icon can have two colors: *yellow* if the sentence is sensitive (positive sensitivity between 0.50 and 0.74) or *red* if the sentence has a positive sensitivity greater than or equal to 0.75. Furthermore, the *"feedback"* column is also shown. This column has two butons which allow the user to send feedback to *Knoxly* so that the tool no longer alerts for sensitive sentences similar to the one for which the feedback is being sent (*Ignore* case) or vice versa (*Good Job!* case). The icons chosen to provide awareness are shown in Figure 5.11. The colors *red* and *yellow* have been used becasue red is the color most used to define a danger/risk, while yellow is the color most used to define the warning/notice. We decided to avoid further highlighting of the sentences in order to do not create cumbersome visualizations and hard to read alerts.

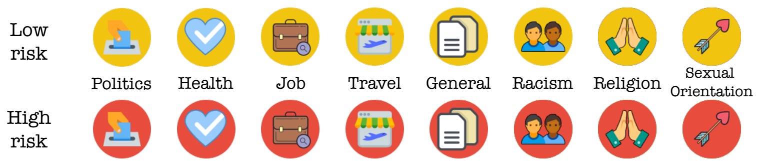


Figure 5.11: Privacy awareness icons adopted in *Knoxly*.

- *Statistics* section (see Figure 5.12) is dedicated to measurements: there is a bar-plot showing the number of occurrences of sensitive sentences that he has disclosed on the Web in the various topics, i.e. politics, health, travel, work and general. Below the bar-plot, the user can see the *overall score*. Such a score measures the sensitivity of the contents that he has disseminated. The overall score *os* is calculated as the arithmetic mean of the sensitivities given to the sentences classified as sensitive by *Knoxly*. The overall score is the online computation of the sensitivity average scores obtained for each of the sentences the user has disseminated, with reference only to the sentences classified as sensitive. The score $os \in [0..1]$ e is represented as a circle that turns "more" *red* as the overall score gets higher; with low overall score the color is mainly *yellow*. Thresholds (for the color changing) have been experimentally chosen, but can be easily fixed once performed a comprehensive usability test [392].

- *About* section (see Figure 5.13) shows information regarding the visual metaphors adopted, with particular focus on the icons used for each topic.

The *Knoxly* UI was created using Bootstrap[13] for the style, and Highcharts[14] for the plots.

[13]https://getbootstrap.com/
[14]https://www.highcharts.com/

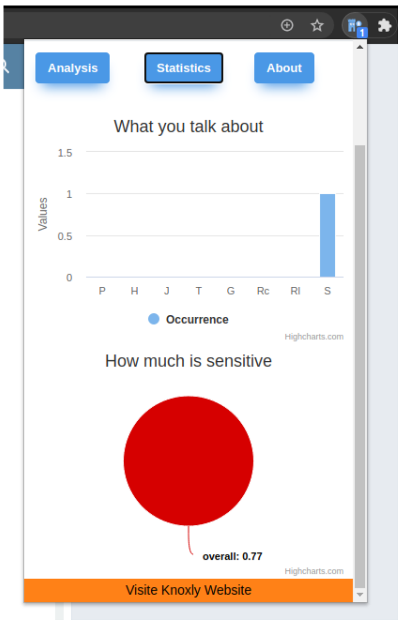
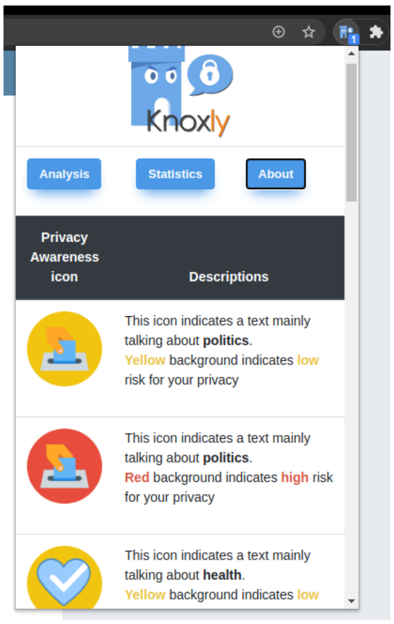


Figure 5.12: User Interface: *Statistics* section.

Keyword module Dictionaries and regular expression within the Keyword module have been saved in the client-side, thus when the user starts the tool the first step is to allocate the module in the main memory.

Customized module This module is stored into Chrome local storage, and consists into eight 514-dimensional arrays (weights), i.e., one for each topic considered in this work. The pseudo-code of the Customized module can be found in algorithm 1. Inputs are the embed of the message m, i.e., em_m, the message topic $topic_m$, and the classes probabilities $sens_m$ returned by the *Knoxly* servers (see Section 5.4.3). The Customized PAII-based model is loaded (line 1) based on the message topic $topic_m$. We then build the multi-view in line 2 which is then input for the prediction made with the Customized PAII-based model (line 3). The final label $label_m$, i.e., -1 for non-sensitive and $+1$ for sensitive is lastly assigned in lines 4; 7.



Privacy Awareness icon	Descriptions
	This icon indicates a text mainly talking about **politics**. Yellow background indicates low risk for your privacy
	This icon indicates a text mainly talking about **politics**. Red background indicates high risk for your privacy
	This icon indicates a text mainly talking about **health**. Yellow background indicates low

Figure 5.13: User Interface: *About* section.

Algorithm 1: *Knoxly Customized module pseudocode*

```
Input  : $em_m$,$topic_m$,$sens_m$
Output: $label_m$
1 Customized-PAII ← CostomizedModule.loadfromLocalStorage($topic_m$);
2 multi-view ← concatenate($em_m$,$sens_m$);
3 $custo_m$ ← dotproduct(Customized-PAII,multi-view);
4 if $custo_m \leq 0$ then
5     $label_m$ ← -1;
6 else
7     $label_m$ ← +1;
8 return $label_m$;
```

5.4.3 *Knoxly*: backend

Knoxly backend is developed in Flask [393], a famous framework to develop server in Python. It consists of a unique end point that once received the client request performs the task 5 explained in Section 5.4.1. The server pseudocode is sketched in algortihm 2. The input is the message m the user is about to disseminate online, the outputs are the embed em_m, the message topic $topic_m$, and the class probabilities for the sensitiveness $sens_m$. In lines 1 and 2 we load the multilingual Universal Sentence Encoder and the Topic classifier, respectively. In line 3 we compute the sentence embedding of m, we pass to the Topic classifier for the topic classification (line 4) and, according to the predicted label the Sensitiveness classifier is loaded (line 5). Finally the classes probabilities are computed (line 6) and the results are returned to the client side.

Algorithm 2: *Knoxly server pseudocode*

```
  Input  : m
  Output: $em_m$ , $topic_m$ , $sens_m$
1 mUSE ← load("https://mUSEurl/");
2 Topic-classifier ← TopicModule.load();
3 $em_m$ ← mUSE.embed(m);
4 $topic_m$ ← Topic-classifier.predict($em_m$);
5 Sensitiveness-classifier ←
   SensitivenessModule.load($topic_m$);
6 $sens_m$ ←
   Sensitiveness-classifier.predictproba($em_m$);
7 return $em_m$ , $topic_m$ , $sens_m$ ;
```

5.5 Experimental study with Knoxly

This section explains how the performance of *Knoxly* have been measured. The performance were evaluated in terms of *efficiency*, that is, how much time, memory and CPU *Knoxly* uses (Section 5.5.1), and *effectiveness*, that is, how accurate the text classification were made (Section 5.5.2) on a dataset composed of “never seen” samples. The first consists into a quantitative analysis, the second into a qualitative analysis. We remark that this is the first study of this kind that measures the effectiveness of the proposed models and systems on a different dataset. The methodology adopted in this phase is sketched in Figure 5.14.

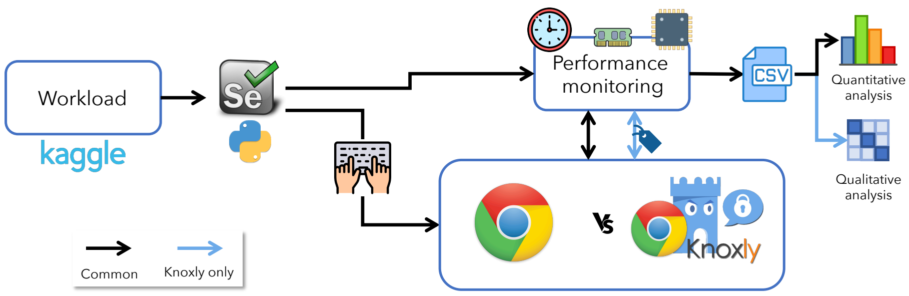


Figure 5.14: Methodology adopted for the experimental study with *Knoxly*.

The qualitative and qualitative analysis was carried out trying to put *Knoxly* as much as possible in a real setting. For this purpose, we employed a new dataset, the `Sentiment140 dataset with 1.6 million tweets`[15], available on kaggle.com. We then selected 1000 entries with a `random` function as workload for the experimental study, discarding those that were shorter than 70 characters so to include only well structured tweets.

We developed a bot which sends the 1000 entries to *Knoxly* exploiting Selenium library for Python. It interacts with a web page and types the sentences. Then, thanks to the input heuristic previously defined (Section 5.4.1) such sentences are sent to the backend. During the sending of the sentences we keep records of the following backend performance: (i) time (in seconds), (ii) RAM used (in Megabytes) and (iii) CPU used (in percentage where each thread used in full corresponds to 100 %[16]). In addition, we record (i) RAM (in Megabytes) and (ii) CPU used (in percentage) for the

[15] `https://www.kaggle.com/kazanova/sentiment140`

[16] A machine with 2 cores, 4 threads can have a maximum CPU consumption of 400%.

Knoxly tool. Observe that we performed the same experiment also without *Knoxly*.

Finally, the results are collected as `.csv` format files and shown in the form of plots.

This experimental study has been conducted using as server a Amazon `EC2 t2.large` machine equipped with `8 GB` RAM, and for the client a MacBook Pro equipped with `Intel Core i5`, `2.7 GHz` and `8 GB` RAM.

5.5.1 Efficiency analysis

The result obtained for the efficiency (quantitative) analysis are shown split between *backend* and *frontend*.

Backend We measure the following performance: (i) *Time*: time means only the time in seconds the server takes to process the request. This measure does not take into account the latency between client and server; (ii) *% CPU usage*: percentage of CPU used by the server to process a request; (iii) *RAM used*: Megabytes of RAM allocated by the server to process a request.

The server typically takes 1.3 to 1.6 seconds to process a request, rarely takes more than 2 seconds and never above 3. The CPU usage is minimum, with almost 90 request out of 100 which demands less than 10% of CPU. The RAM allocated ranges between 10 and 12 Mb, with up to 90% of requests which needs less than 12 Mb to be processed.

Frontend We set-up two configuration: *Knoxly* and *noaddons*. The former consists in Google Chrome with *Knoxly* installed, the latter consists in plain Google Chrome without any extension. We measure the following performance: (i) *% CPU usage*: percentage of CPU used by configuration to process a request; (ii) *RAM used*: Megabytes of RAM allocated by the configuration to process a request.

Figure 5.15 shows the cumulative distribution function concerning the two configurations CPU usage while processing a re-

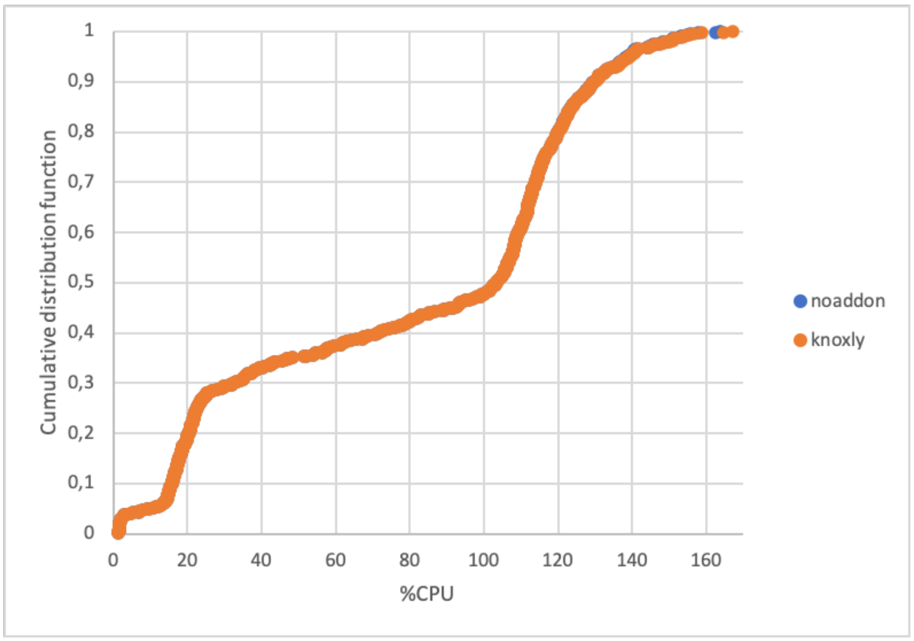


Figure 5.15: Comparison between the Cumulative Distribution Functions for the CPU usage of *Knoxly* and *noaddons* configuration.

quest. We observe that *Knoxly* configuration's CPU usage trend is on par with *noaddons*. Both with or without the extension we use 160% of CPU. Up to 90% of requests takes less than 140% of CPU.

Figure 5.16 shows the cumulative distribution function concerning the two configurations RAM usage while processing a request. We observe that initially *Knoxly* configuration allocates more RAM than *noaddons*: this is due to the loading into the main memory of the *Keyword module* and the *Customized module*. The other allocated resources are due the User Interface, which uploads the privacy awareness icons for each message analyzed and classified as sensitive.

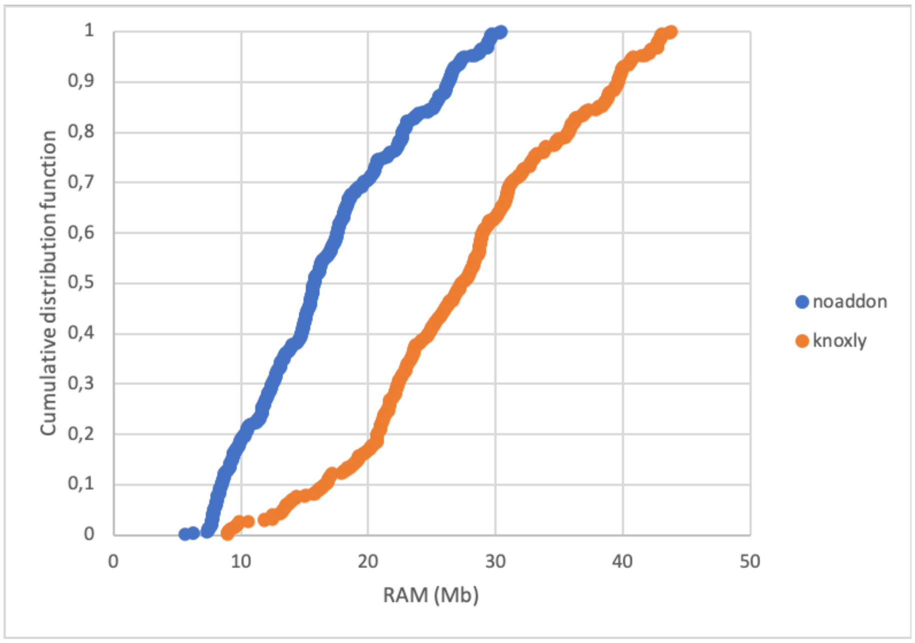


Figure 5.16: Comparison between the Cumulative Distribution Functions for the RAM allocation of *Knoxly* and *noaddons* configuration.

5.5.2 Effectiveness analysis

This analysis was carried out to assess how well the classifiers made are capable of making correct predictions on a completely different set of data. To keep track of the predictions made by the classifiers we employed three expert annotators. We measured their inter agreement with Fleiss' Kappa, obtaining $k = 0.63$, that is a "substantial agreement" (closer to "moderate agreement"). Such experts annotated the texts in terms of topic and sensitiveness. Then, we compared such annotations against the classification results obtained by the *Topic module* and the *Sensitiveness module*. The main information on the dataset used are summarized in Table 5.14.

Topic	**# samples**
Politics	3
Health	148
Job	61
Travel	128
General	560
racism	8
Religion	8
Sexual Orientation	42

Table 5.14: Information on the dataset used for the evaluation of *Knoxly* in real-world.

Effectiveness of the Topic module We aim at understanding whether the topic classification performed by *Knoxly* is correct or not.

		`Predicted label`							
		P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
`True label`	**P**	**3**	0	0	0	0	0	0	0
	H	0	**99**	0	18	1	25	4	1
	J	0	8	**10**	30	0	9	1	3
	T	2	5	3	**94**	4	20	0	0
	G	7	21	22	*200*	**193**	100	6	17
	Rac	1	0	0	1	0	**6**	0	0
	Rel	0	0	0	1	0	1	**6**	0
	SO	0	2	0	5	1	9	0	**25**

Table 5.15: Confusion matrix for the *Knoxly Topic module*. In *italic* the worst case. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = racism, Rel = Religion, SO = Sexual Orientation.

Table 5.15 shows the confusion matrix obtained in the effectiveness/qualitative analysis. The average F1-score varies from 0.52 to 0.72 according to the specific topic class, except for "racism" and "job". We can observe that the *Topic module* makes the

greatest number of correct predictions on topic "health", while, it often misclassifies a sentence belonging to the topic "general" as "travel". This is due to the dataset used for training: *(a)* the case of "travel" is made of tweets posted on Twitter, more morpho-syntactically similar to those texts used in this experimental study, *(b)* in the case of "general" it is a more detailed description of a film plot, somewhat dissimilar to the texts used in this test.
With regards to the misclassification involving the "racism" topic, by a closer inspection, we observe that such behavior happens due to insulting, bad words, cursing in the tweets which are common words used in racism posts. For example, the tweet "*said bye to my deadly bitch nasty girl nade. gonna miss her but she's working at myer here so still see her heaps*" which has been annotated as belonging to "sexual orientation", has been classified as "racism". Concerning the errors made classifying as "health" a message which belongs to "general", we notice that the issue is focused on the use of metaphors such as in the tweet "*how could you mention Bill Hicks while I'm sitting here listeneing to the world's saddest song? Now I need a nurse*" where the user has expressed the need of a nurse, which is a medical/health concept.
During this analysis we also noticed that "politics" topic is hard to classify. The political landscape may vary in a short time (politicians, politic party, etc.) and it is different from country to country, from region to region, therefore to obtain a high-quality classification our system should account a load of contextual information. For example, we have found the tweet "*voting for my favorites teens choice awards. i hope will win and Vanessa Hudgens too*" which belongs to "general", but classified as "politics".
Overall, we observed that there are some tweets which are hard to classify as belonging to one only class. Indeed, we have found, for instance, several samples that are about "job" but also relate to "travel" because disclosing a shift from a certain location to some others or that they are located in a specific place. Examples are in the following: (i) "*hey everyone, just got off work. glad to be home after getting poured on for 6 hours*", (ii) "*Back home after my test for maths made absolutely NO SENSE! Horrible!*", (iii) "*If*

I can stand at the concert I will take photos to share with you guys. Now I have to vomit again! Sry Goodbye". And this holds true for the case of "health" messages classified as "travel" and "job" classified as "health" (where people complain about their stress or other health conditions due to their job). This provided precious insights for future works, moving from a multi-class problem to a multi-label problem.

The primary cause of misclassification and errors is on the "general" topic. This happens due to its broad nature, which covers an endless number of domains, from weather conditions to purchases, from festive wishes to the story of what the user liked about a radio show. Such nature cannot be accounted with only movie plots, as we did, but needs further diverse datasets to be processed.
Since related works in literature have not considered the case "general" (see Section 5.6), to better assess the effectiveness and performance of Topic module so to compare with previous initiatives on the issue, we need to drop the "general" topic from our model. The newly trained model's performance are on par with those observed in Section 5.3 with F1-score above 0.97 for each class. We then embedded this new model into *Knoxly* and re-processed the dataset used for the evaluation phase without the samples belonging to the "general" class. In this setting *Knoxly* exhibits an average F1-score of 0.83 for all the topics. The majority of errors happens for tweets that can be categorized as belonging to multiple topics.

Effectiveness of the Sensitiveness module We aim at understanding whether the sensitiveness classification performed by *Knoxly* is correct or not.

Table 5.16 shows the result of this effectiveness analysis on the *Sensitiveness module*. It can be seen that the number of false positives is greater than the number of false negatives. This means that the *Sensitiveness module* classifies and reports more sensitive content than it should. This could be assumed as a "fake problem", since ideally it should be better to have a classifier that alerts

		Pred. label			
		All		Correct topic	
		ns	**s**	**ns**	**s**
True label	**ns**	359	324	191	63
	s	74	209	55	127

Table 5.16: Confusion matrix for the *Knoxly Sensitiveness module*. In *italic* the worst case. *s* = sensitive, *ns* = non sensitive. *All* = classification made on all samples, *Correct topic* = classification made on only samples correctly classified by Topic module.

a non-sensitive sample as sensitive and not vice versa, otherwise sensitive and/or personal data would risk in being disclosed on the Web. However, the user can update the knowledge of *Knoxly* (*feedback* button) to adapt the alerts/warnings on his own perception of privacy, thus decreasing the number of false positives. Large part of the errors have been made because of the misclassification in the Topic module which has spillovers on the Sensitiveness module as well (see Section 5.4.1). Indeed, if we take apart misclassified samples with regard to the topic (see right side of Table 5.16), we observe that the sensitiveness module exhibits superior performance, shifting from an average F1-score of 0.60 to 0.73.

Effectiveness of Customized module In order to test the *Customized module*, we had to perform a slightly different experiment. We need here to simulate a user which does not want to be warned against some contents. Through Selenium we have automated (i) the typing of natural language texts and (ii) the *feedback* button tapping, in particular “Ignore”. The idea is preparing a dataset of texts that are sensitive and to classify them with the *Customized module*: it will provide high accuracy. For each text, we want to send *Knoxly* a negative feedback and repeat the classification of the entire dataset. We expect that if the module learns correctly from the user’s feedback, then the accuracy resulting from the classification decreases as more negative feedback we send.

For this experiment we used a minimal dataset, consisting of 10 sensitive elements for each topic (politics, health, job, travel, general, racism, religion, sexual orientation). We have augmented the size of this dataset by 200% by populating it with sentences having a similar meaning but written in other languages (Italian in particular) or using synonyms. For the former case we used deepl[17], a famous deep learning-based translation tool, for the latter, we employed Reverso[18], an online service that helps users in domain-specific translations and searching for synonyms. In Table 5.17 there is an example of this task for the topic "travel". Thus, we will send 30 feedback, one for each text written.

Original sentence	Not sure what you are talking about. She is going on nonstop flights SNA to SFO and then SFO to EWR
Reverso	Not sure what you are speaking about. She is riding on nonstop flights SNA to SFO and then SFO to EWR
deepl	Non so di cosa stai parlando. Lei sta andando su voli nonstop da SNA a SFO e poi SFO a EWR

Table 5.17: Example of data augmentation for the Travel dataset in the effectiveness experiment for the *Customized module*. *Reverso*: sentence refactoring using the famous online service, *deepl*: translation of the sentence from English into Italian language.

The experiment's result is encouraging as all classifiers decrease their performance as the user (bot selenium in this case) sends negative feedback. In general all classifiers start with 100% accuracy, slowly decreasing the performance as more feedback we send, until reaching accuracies in the range 75% to 80%. We observe that, in the case of health, a small number of feedback is enough to improve the knowledge of the classifier and to adapt its classification to the user's perception of privacy. Indeed, in this case the accuracy decreases gradually to reach 75%. For the other

[17]https://www.deepl.com/translator

[18]https://synonyms.reverso.net/synonym/

classifiers, instead, the user needs to send more feedback before the classifier understands his attitudes to privacy and adapts the warnings/alerts.

5.6 Related work

In the recent past, several attempts have been made to protect the users' privacy on the Web. The vast majority of the works focused on protecting users' personal attributes or identities. It is plenty of them, we can mention [394, 395, 345] working on privacy of social media users, some authors focused on age-based, and inactivity-based withdrawal systems [396], and in [336, 337] repair strategies for regret Facebook posts/tweets have been identified.

However, only a few works have focused on the classification of sensitive, personal or inappropriate content disseminated on the Web. This phenomenon is a real threat for users due to inadvertent or unintentional disclosure during socialization promoted by social media. Privacy-enhancing techniques such as l-diversity [397], t-closeness [398] and differential privacy [395] have been developed to sanitize the dataset before publishing. They have shown to be vulnerable against several attacks, e.g., [399, 400], and more importantly, they are not feasible in online socialization, since friends are allowed to access profile, posts, etc. Also, access control frameworks such as Persona [401] and EASiER' [402] have been developed. However, these systems require the user to explicitly define what is private and what type of protection he needs. In [359], authors have studied Twitter users that unwittingly post sensitive information about themselves and other people for whom there may be negative consequences. They mainly analyzed three types of leaks, i.e., divulging vacation plans, tweeting under the influence of alcohol, and revealing health conditions. Authors considered various lexical and syntactical patterns for all three topics but also location-related information for the case of "vacation plans" (manually annotated), to extract the tweet's topic and applied machine learning techniques, in particular a Support Vector Machine and Naive Bayes classifiers, to classify those tweets. They obtained accuracy from 0.79 to 0.90 according to the topic. Compared to this work we accounted more topics, indeed we also included "job", "politics", "racism", "religion", "sexual orientation", and "general". This last topic is useful because we cannot rely on a system that always classifies texts into one of the more sensitive categories like [359]. Furthermore, the authors' system is mainly based on lexical features with the well-known limitations (e.g., maintenance, semantics not considered) while our system accounts for both lexical and semantics. Lastly, our solution has been embedded in Knoxly, a Google Chrome extension prototype, which can be used by the end-user and has undergone an evaluation phase, in contrast to the proof-validation ("in silico") of the method done in [359].

In [175], the authors propose a system for automatic assessment of privacy disclosure, classifying 500 tweets manually annotated as sensitive or not (i.e., binary classification). They compared RF, SVM and NB methods finding an average F1-score from 0.53 to 0.68 according to the ML method adopted. The features engineered are not clear. Later on [178], the authors employed Amazon Mechanical Turk workers to annotate a larger corpus of tweets (6000) with no hashtags on a three-level scale of privacy. They represented the tweets with word-based TF-IDF. They obtained F1-score from 0.53 to 0.69 according to the ML method employed for the classification. Both works, differently from us, did not account the topic of the message and developed a tool for increasing final users' awareness with respect to their privacy.

In [182], the authors adopt an inverse strategy compared to our work. They first propose a system to understand if the content of a tweet is private or not, then if the content is private, it is categorized on several topics like us. They employed Amazon Mechanical Turk workers for the annotation process of Twitter messages obtaining a moderately acceptable agreement. Then they applied the RF method for both tasks (sensitiveness and topic classification) consecutively. Their proposal exhibits from 0.79 to 0.88 F1-score in validation, but has not encompassed a testing phase or evaluation on a different dataset.

Recently, Wang et al. [21] proposed a context-aware, text-based quantitative model for private information assessment, namely PrivScore. First, they built a context-free scoring system: they examined users' (Amazon Mechanical Turk workers) opinions on the levels of contents' sensitiveness, and then build a semantic model that comprehends the opinions to generate a context-free PrivScore. The model is primarily based on deep learning (LSTM network) and word embedding (GloVe [403] in particular) and produces the so-called PrivScore with an average precision and recall of 0.85. Then authors developed a context-aware PrivScore that measures the influence of societal context using the volume, duration, and relevance of trending topics. When the election period starts, many tweets are about politics; thus, the degree of sensitiveness of political content implicitly decreases from the non-election days. Lastly, the authors envisioned a personalized privacy score that should be based on the users' tweets history and integrated with an AI-based bot. Such a last model has not been implemented. In contrast with this work, we did not rely on AMT because we wanted to develop a system compliant with the GDPR guidelines and capable of adapting to the users' privacy attitudes only when given to the end-user, not based on annotation done by non-expert persons. In our work, we did not use deep learning but an ensemble method, i.e., Random Forest, and we feed such a model with sentence embeddings rather than word embeddings. All the modules designed and presented in this paper have been developed and embedded into Knoxly extension, in contrast to [21], where authors only defined/designed the personalized score and performed a proof-validation for only part of the whole solution.
More recently, authors in [187] proposed a fuzzy-rule based system to score privacy of tweets based on uni-gram and positional bi-gram. The fuzzy system is based on the concept of *background knowledge*, that is it considers any structured information about the users (also from different social media) to create a context. They also model the problem on a three-level scale of privacy

(as [178]), obtaining F1-score from 0.35 to 0.73 on the different experiments and settings of ML methods employed.

Another recent project classified private tweets into a set of pre-defined categories, using BoW/TFIDF features and machine learning, specifically a Naive Bayes classifier [404, 405]. Authors assumed that sensitive tweets are pre-identified, without explaining how that could be achieved. Their technique to extract term-based features cannot capture semantic features or accurately discover topics containing subtle yet sensitive content.

In Table 5.18, we show the comparison between the results of our work and those obtained by previous initiatives in literature. We compare on different criteria as it follows: (i) *Topic* – has the topic been accounted?, (ii) *Sens.* – has the sensitiveness been accounted? And how the problem was formulated? Binary or differently? (iii) *Custom.* – has the work developed a customized model? (iv) *Type* – has the work developed a tool or is it a study?, (v) *Real-world eval.* – has the work performed a real-world evaluation (on a different dataset) of the proposed method/system?, (vi) *Perform. (avg. F1)* – overall performance achieved by the proposal in terms of avg. F1-score.

	Topic	**Sens.**	**Custom.**	**Type**	**Real-world eval**	**Perform. (avg. F1)**	
Our work	✓(GDPR)	✓(binary)	✓	Tool	✓	*In silico* *In real-world*	0.90 0.60
[359]	✓(3)	✓(binary)	✗	Study	✗	✗	0.79
[175]	✗	✓(binary)	✗	Study	✗	✗	0.59
[178]	✗	✓(3 levels)	✗	Tool	✓	✗	0.60
[182]	✓(GDPR)	✓(binary)	✗	Study	✗	✗	0.84
[21]	✗	✓	✗	Study	✗	✗	0.85
[187]	✗	✓(3 levels)	✗	Study	✗	✗	0.54

Table 5.18: Detailed comparison of our work against others proposed in literature.

As we can see from Table 5.18, our proposal's performance in the "in silico" setting overcome state-of-the-art solutions. Concerning the performance "in real-world" we observe that they are on par with works accounting both topic and sensitiveness, and below the proposal which accounted only the sensitiveness. Indeed, as we showed in Section 5.5, when we only consider correctly topic-classified samples, the performance of sensitiveness module increase. Instead, we remark that our performance “in real-world” are lower than [359] due to the larger number of topics considered.

To the best of our knowledge, this is the first work that developed and evaluated a prototype for the end-user, which provides privacy safeguards through awareness mechanisms; such mechanisms are capable of alerting the user before disseminating sensitive content on the Web.

5.7 Discussion

5.7.1 Limitations

Knoxly plugin is currently a prototype, therefore works on four categories of Websites: mail services (such as gmail.com), messaging apps (such as telegram.com), microblogging platforms (such as twitter.com), and blogging platforms (such as reddit.com). Its compatibility with the Website must be manually verified. This due to (a) discrepancies in the input/text areas of the User Interfaces across multiple social media or websites, and (b) the tool behavior could be assumed as risky for some online service and being blocked consequently (this is the case of facebook.com).

Currently, *Knoxly* provides mainly privacy awareness alerting the user before sending a message with sensitive and or personal contents but does not provide contingency mechanisms except a customized version of the well-known k-anonimity algorithm which substitutes k characters of a sensitive/personal word with “*”.

The conducted evaluation showed how the tool performance is on par with those obtained in literature by related works. Nevertheless, there are several margins for improvements.

We remark that we aimed to lay the foundations of a system to provide the users with the ability to control the dissemination of (private) information, assessing its feasibility to head towards a more complex and effective system in the future. The datasets used here can be considered minimal and the manual annotation task, if employing more experts in the privacy field, could have resulted in more data that would favor better results with all the modules. To deal with this limitation, drawing on the perspective of citizens science [406], we have set up the official Website of *Knoxly*, `https://knoxly-website.herokuapp.com`, in which we invite knowledgeable persons to collaborate in the labeling process. We aim to gather as much labeled samples as possible to improve the *Knoxly* heuristics. As also emphasized in the literature, we will measure the Inter-Annotator Agreement, that is how well multiple annotators can make the same annotation decision for a particular category (e.g., employing the Cohen's kappa coefficient), and we will examine the responses also to derive a perceptual model behind the privacy decisions expressed.

5.7.2 Conclusion

Although much attention has been devoted to the protection of personal identities and attributes on the Web and social media, very few works have tackled the issue here presented, i.e., the unaware dissemination of sensitive information in the form of text messages. To the best of our knowledge, such systems have only been conceived but not implemented as a tool for end-user like we did. Indeed, to address this problem we designed, implemented and evaluated a system (then embedded in a Google Chrome extension, named *Knoxly*) which is capable of classifying sensitive contents in a message written in natural language, alerting the user before the dissemination of such message, and adapting to his attitudes towards privacy online. We showed the feasibility of this kind of solution both "in silico" and "in vivo" settings (using Selenium WebDriver), and we obtained promising results. With "in silico" testing phases we obtained up to 99% F1-score in classifying the topic of a message, up to 88% accuracy in classifying the sensitiveness of the message's contents, and up to 98% accuracy when using the customized approach. With "in vivo" evaluation, we obtained excellent performance in efficiency, always less than 50Mb RAM and negligible CPU usage, and effectiveness comparable with other related works in literature.

5.7.3 Future works

Given the encouraging preliminary results about the use of automated approaches to safeguards users privacy, we are currently working on how to mature the prototype presented here for its widespread usage through the Chrome Web store. At the same time, we aim to enlarge the datasets used in terms of size, hence the use of `https://knoxly-website.herokuapp.com`, the official Website for *Knoxly* project. Regarding the size, we need to involve more experts in the privacy field to manually annotate the data because we cannot rely upon AMT workers (due to the lack of expertise), which should be considered for a comprehensive user evaluation [25, 392]. For what concerns a contingency mechanism, we aim to design an AI-based solution to suggest messages modifications in addition to the basic k-anonymity mechanisms already delivered.

Chapter 6

Expanding research scope: ML for child protection

Chapter Highlights

- Proposal of a novel techno-regulatory approach based on ML to uphold legal standards for online child protection; the approach exploits the analysis of touch gesture to distinguish the age of a mobile device users;
- Investigate the use of multiple touch gestures in combination to perform the classification between adults and underages;
- Evaluation of the effectiveness of proposed approach on a large dataset obtaining high accuracy score in classifying between underages and adults;
- Classification according to the taxonomy depicted in Chapter 2: **{ Type of work**: Tool, **Threat**: illegal/inappropriate conducts, contacts and contents, **ML Method**: Fusion models, **Type of protection**: n/a **}**

6.1 Introduction

According to a 2018 Pew Research study [407], smartphones and social media are now an almost universal feature of teenage life in the United States, with more than nine-in-ten U.S. teens ages 13 to 17 accessing them. Apart from advantages and opportunities, the Internet exposes numerous threats for children/teenagers: from access to inappropriate content to exposition to dangerous behaviors. The same study also states that parents use a wide array of strategies to monitor their teens' technology use, including 52% of parents who install parental control applications on their teens' mobile devices to filter and block inappropriate online activities.

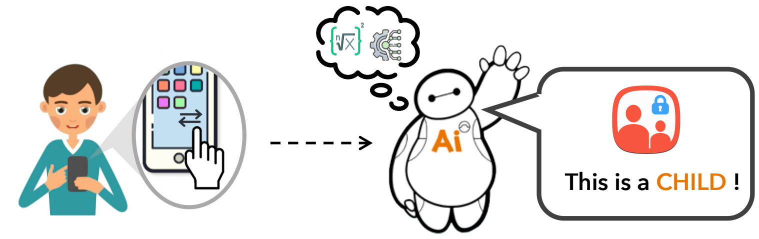


Figure 6.1: A general overview of the proposed system (AI4Children) for smartphones' users age classification.

In this section, we present a novel approach aiming to protect children when interacting with smartphones (a sketch of the proposed system is available in Figure 6.1). Specifically, we face the problem of classifying mobile users into two groups: *underages* and *adults*. The *age-threshold* used to distinguish between the two types of subjects is 16, as stated by the art. 8 of the *EU's General Data Protection Regulation*[1].

Touch gestures such as swipes, taps, and keystrokes, are common modes of interaction with smart touchscreen-enabled devices [408]. Major platforms including Android OS and iOS provide a variety of APIs to help developers detect gestures aiming to enhance apps' quality of experience. Access to these APIs allows apps to collect raw gesture data from different sensors available on the smart device. The fine-grained nature of this data became appealing for research, indeed touch gestures have been used for person recognition [409], for user authentication when combined with sensor data [410], and for other applications such as the complex task of bio-cryptography [411] and to foster social

[1] http://www.privacy-regulation.eu/it/8.htm

communication in children with autism [412, 413]. Thus, by starting with the observation that underages and adults perform commonly used touch-gestures in a different way on mobile devices, we developed an Android app in order to collect, in an experimental study involving 147 participants, more than 9000 heterogeneous touch-based gestures. We carried out several experiments to find, by exploiting ML techniques like in [285, 219, 414, 415, 409], the best combination of touch-based gestures capable of distinguishing between adults and underages.

The main *contributions* of our work can be summarized as follows:

- Proposal of a novel techno-regulatory approach exploiting ML techniques to provide safeguards against threats online. We study a set of touch-based gestures, to determine whether it is possible to distinguish who is accessing a smartphone, i.e., underage or adult, to guarantee protection.

- Evaluation of the effectiveness of our approach, on a large dataset including more than 9000 touch-gestures from 147 participants. We experimented both *single-view* and *multi-view* learning techniques to find the best combination of touch-gestures capable of distinguishing between adults and underages. Results show that the multi-view learning combining just three touch gestures, that is, scrolls, swipes, and pinch-to-zoom gestures, achieves the best ROC AUC (0.92) and accuracy (88%) scores.

- Several improvements against related works available in the literature: *(i)* rely on three touch gestures with fewer features to compute (just ten features for each gesture), *(ii)* consider some relevant multi-touch gestures, such as the pinch-to-zoom, which as proven, carry loads of information, *(iii)* experiment our approach on different types of smartphones.

The rest of the section is organized as follows. In Section 6.2, we describe some relevant works in the field of child protection and user identification based on the analysis of touch-based gestures performed on mobile devices. In Section 6.3, we provide an overview of the motivations of our research. In Section 6.4, we describe our approach to distinguish between underages and adults when analyzing touch-based gestures. In Section 6.5, we discuss the results and, finally, in Section 6.6 we conclude with some future directions.

6.2 Related Work

Despite of several advantages and opportunities discussed in the previous section, Internet exposes children/teenagers to numerous threats, ranging from access to inappropriate content to exposition to dangerous behaviors. Several child protection systems, categorized as parental control systems, provide parents with numerous instruments to protect children from such threats [2].

In recent years, several works [416, 417, 418] proposed systems for continuous authentication primarily based on data streams coming from gyroscope and accelerometer sensors. They integrate different techniques of deep learning, such as Convolutional Neural Networks and features fusion, obtaining encouraging results in users recognition. Differently from these works, we used only the information extracted from touch gesture data.

Other works available in literature are based on the analysis of touch-based gestures performed on smartphones, arguing that user information could be extracted and used to control smartphones' interactions. This information could improperly be used to track users and distinguish between them to provide access and functionalities diversified on a per-user basis.

In [419], the authors describe a novel multi-touch gesture based authentication technique, defined on a set of five-finger touch gestures. The authors built a classifier to recognize unique biometric gesture characteristics of an individual, achieving an accuracy

[2]For the under 16s, see the *Family Link*`https://families.google.com/familylink/`

rate of 90% with single gestures. In [420], the authors analyzed a set of 30 behavioral touch features, that can be extracted from raw touchscreen logs, and demonstrate that different users populate distinct subspaces of this feature space. The authors collected touch data from users interacting with a smartphone by using *only* up-down and left-right scroll gestures. They proposed a classification framework that after learning the touch-based user behavior, can proceed by accepting or rejecting the current user. In [421], the authors present *SilentSense*, a framework to authenticate smartphone users. The main idea is to exploit biometrics information obtained from the touches and leverage on the sensors to capture the device's micro-movements caused by user's actions. Conversely to the described approaches whose main goal is to preserve security on smartphones by avoiding intruders' access, our main goal is to recognize a specific category of users and adapt behaviors accordingly.

In [422], authors argued that touch-based gestures on touchscreen devices constitute a privacy threat. They show how the combination of swipe, tap, and handwriting gestures reveals up to 98.5% of information about users. It is worth noting that as explained in [423, 424], handwriting is not one of the most common touch gestures, and in literature, several studies acknowledge handwriting as a biometric [425]. In addition, the experiment they performed involved 89 participants but only 30 of them used all the envisioned games and hence provided samples for all gestures. Participants were free to join or leave the experimental phase whenever they wanted; therefore, the number of performed touch gestures considerably varied from participant to participant; conversely, in our study, all participants interacted with all games and therefore provided samples for all gestures.

In [426], authors present a technique to classify the users' age group from touch gestures. In their work, a child is a person having 6 years at most, and the dataset collected from 119 participants (89 children ages 3 to 6) included 587 samples. Using a Bayes' rule classifier, their technique delivered 86.5% accuracy when classifying each touch event one at a time, and 99% accuracy with a window of 7 or more consecutive taps. Differently from our work, the authors analyzed a dataset composed of instances associated with actions performed by children who are actually very young (3 to 6 years old). With these data, therefore, is very easy to classify individuals correctly as children have a very different tactile behavior compared to that of adults. Furthermore, it is clear that children's input performance and touch accuracy improve with age [427]. In our study the collected dataset included 9983 touch-based gestures among which 2942 gestures were taps. Our evaluation phase involved 147 participants with a better age distribution, and more than 30% of the participants were in the range 7-16 years old while more than 25% in the range 17-21. Finally, in our work, we do not consider touch gesture windows but combinations of single gestures performed by the same participant.

In another similar work [428], the authors present techniques to detect whether a child is on a mobile phone. They analyzed touch-based gestures as well as sensors features (and a combination of thereof). Fifty subjects (25 children and 25 adults) were recruited, with a clear gap between ages of underage (range 3-12) and adults (range 24-66). They evaluated the Random Forest classifier when using tap and stroke features and when bundling multiple gestures together. The results show good performance on the age group detection task with over 0.99 AUC for all the three approaches investigated. We differ from this work in several ways. First of all, our sample was larger, with 147 individuals and with a better age distribution. Second, we investigated a multi-view learning approach for the age classification problem, while authors in [428], clearly state that they had *"not examined other types of gestures like multi-finger gestures and the possibility of fusing different classifiers of different gestures for better and faster detection"*. Finally, they had not evaluated their models across different devices and different vendors.

6.3 Child protection: why?

Beyond motivational issues, one thing that clearly emerges by the related works is the need for cross-disciplinary experimental activities leading to more and more efficient techno-regulatory approaches binding regulatory priorities with opportunities provided by technological safeguards. From this perspective, experiments appear to be essential as there is still a lack of the expertise, the experiences, and the hybrid skills (computer science, law, interaction design) needed to achieve adequate regulatory results from all points of view (formal compliance with existing legal standards, effectiveness, scalability and technical feasibility). In this direction, we propose a techno regulated-based approach that exploits ML techniques to classify individuals, specifically underages, while they interact with technology, with the final goal of protecting them against specific online threats.

The explosion of information and communication technology, as emphasized in the 2015 edition of Guidelines for Industry on Child Online Protection released by International Telecommunication Union (ITU)[3] and by UNICEF, has created unprecedented opportunities for children and young people to communicate and access information but, at the same time, significant challenges to children's safety [429]. Online threats are numerous [430]: cyberbullying, grooming, hidden advertising, non-illicit contents that are still harmful to psychological well-being, and kiddie porn material.

The latest Google Transparency Report[4] gives a rough idea of the scale of issues at stake. In the period from July to September 2018, about 1.7 million videos (more than the 22% of the whole number of the videos removed) have been removed just from YouTube because unsuitable for children (videos containing adult themes, nudity, violence). Despite that, there is still some danger because contents must be removed proactively or in real-time with an *ad-hoc* and customized solution: indeed, more than 25% of the

[3] `https://www.itu.int/en/cop/Pages/default.aspx`

[4] `https://transparencyreport.google.com/netzdg/youtube`

content has been deleted after at least one visualization.

For children risks are even higher since they often circumvent or uninstall parental controls by lying about their age. At the same time, parents do not always understand the potential risks their children may encounter since they often underestimate teenagers' exposure to sexual content or overestimate it due to mass media messages [431, 432]. Against this background, it is easy figuring out how the issue led to national and international initiatives and regulatory actions, among which the most recent is the GDPR[5], which limits the contents that can be shown to 13 to 15-year-old users. What emerges is that alongside traditional protections, it is necessary to develop other types of protection able to hinder use by protected parties.

6.4 Our approach for age detection

In this section we describe the approach we proposed for the classification of mobile users in *underages* or *adults*. The approach, as proposed in [285] and shown in Fig. 6.2, envisions different phases to *(i)* collect data, *(ii)* build the data sets through feature extraction and data labeling, *(iii)* apply ML methods to derive the best combination of gestures and the best ML technique for the age group classification.

6.4.1 Phase 1: Data collection

In order to collect data, we implemented an Android app that allows to capture and analyze user interactions with a mobile device. Such app, named *Artificial Intelligence for Children* (*AI4C app*[6]), is essentially a simple game consisting of a series of tests, or *micro-games*. Each micro-game allows to capture a specific type of touch gesture. According to [423, 424], we consider the

[5] `https://gdpr-info.eu`
[6] `https://bit.ly/2M4sE2G`

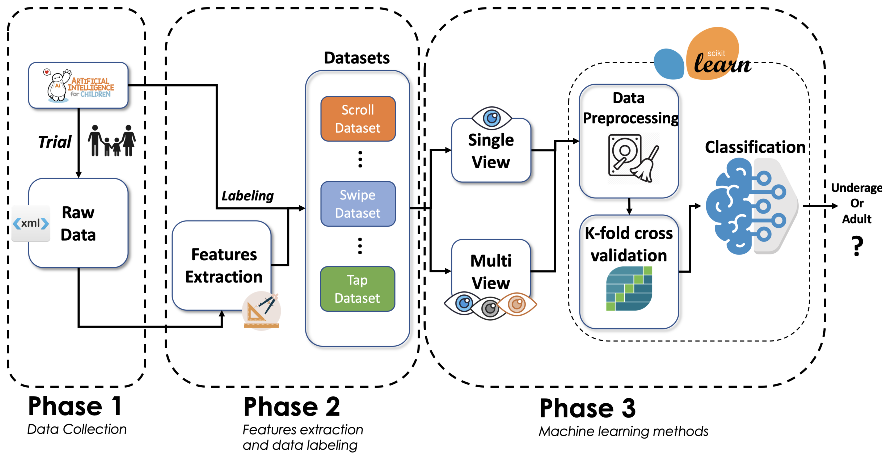


Figure 6.2: A sketch of the phases envisioned in the proposed approach for the age detection. Phase 1: data collection, through the use of *AI4C* app. Phase 2: feature extraction, labeling an dataset building. Phase 3: application of ML methods (single and multi-view) and results.

following touch-based gestures: *scroll*, *swipe*, *tap*, *drag & drop*, *pinch-to-zoom*.

AI4C app provides the following micro-games:

- *Reading (scroll gestures):* it allows to read a Disney cartoon composed of a sequence of 6 pages, that have to be scrolled down (see Fig. 6.3(a)).
- *Candy Pacman (swipe gestures):* it has been implemented to capture lateral swipes gestures performed to move Pacman and allow it to eat a candy (see Fig. 6.3(b)).
- *Color Matching (tap gestures):* it shows a single color on the top of the window and a grid of colors on the bottom. When a color appears on the top, the user has to select the same color, by tapping it on the grid (see Fig. 6.3(c)).
- *Score a goal (drag & drop gestures):* it allows the user to select the ballon, positioned in a random position on the

screen, and drag it to score a goal (see Fig. 6.3(d)).

- *Writing (keystroke gestures):* a short sentences from Disney cartoons is displayed on the top of the screen and the user has to write the same text at the bottom; it has been implemented to capture keystrokes events performed by the user while writing on the keyboard (see Fig. 6.3(e)).

- *Calculation (Pinch to zoom):* it shows a blue rectangle with a small text inside. To read the text, the user has to pinch and zoom. The text is an arithmetic operation, for which the user has to write the right solution at the bottom of the screen (see Fig. 6.3(f)).

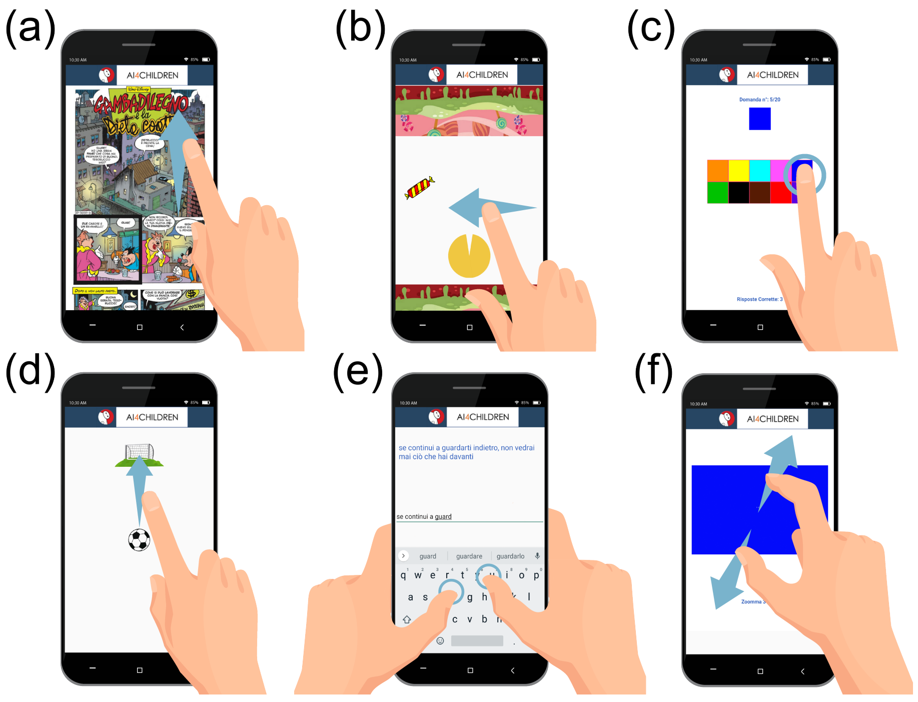


Figure 6.3: The six micro-games of AI4C app: *(a) Reading, (b) Candy Pacman, (c) Color Matching, (d) Score a goal, (e) Writing, (f) Calculation.*

In Table 6.1 we provide details about the information we captured for each analyzed gesture when executing the 6 micro-games provided by our app.

Before starting the collection phase we explained to subjects what they were expect to do in the study. For children, their parents completed written parental permission forms. All participants had to sign consent forms. We explained that we did not collected personal information during the game process except for username, device ID, and age. Raw data associated to gestures performed by users are tracked and saved for the subsequent analysis. We also explained that this data were kept confidential and used only for the period of the experimentation. Moreover, subjects used smartphones provided by us, in order to avoid to use personal devices and to be sure to use devices with the characteristics needed in our study.

Data collection, done through the *AI4C* app was mainly conducted at the “University of Salerno” and in collaboration with the “Gino Landolfi Primary School” in Agropoli (SA). The models of smartphones used for the experiments were an LG Nexus 5X, an ASUS ZenFone 2, and a HTC Desire 820. Data were collected from 147 participants, with the age distribution shown in Fig. 6.4. The age varies in the range from 7 to 59 years. The sample was composed of 46 underages and 101 adults. 80% of adults falls into the range 16 - 29 and 20% into the age range 30 - 59. Moreover, 61% of the participants were males and 39% females.

6.4.2 Phase 2: Features extraction and data labeling

The aim of this phase is to identify significant features within the raw data, and therefore build the data sets that will be used by the ML algorithms tested during the classification phase (Section 6.4.3). As explained before, such data sets were labeled (0 for underages and 1 for adults) according to the *age-threshold* used to distinguish adults and underages by the GDPR and whose value is 16. Table 6.2 shows the data sets generated for each type of

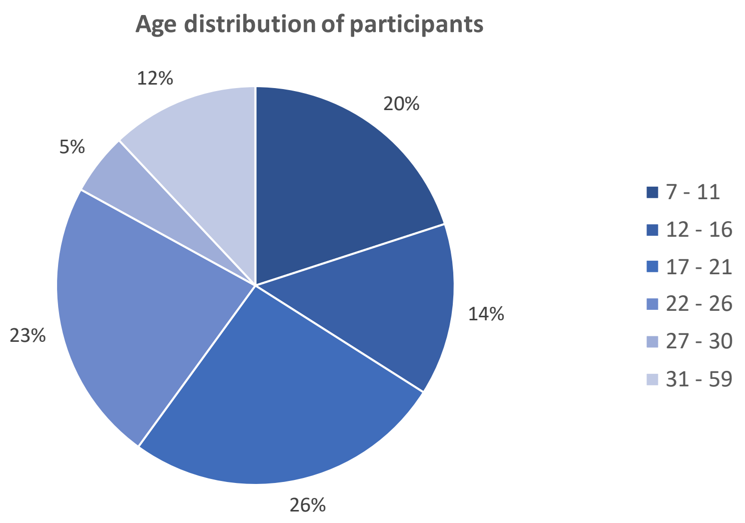


Figure 6.4: Age distribution of the participants at the study. 34% of participants was under the threshold of 16 years.

gesture, with the number of calculated features.

Scroll down data set. As shown in Table 6.2, we collected 2594 scroll down gestures, where 31% were performed by underages and 69% by adults. To build this data set, for each scroll down gesture we calculated 50 features. As we can see in Table 6.3, there exist several types of features. Some of these features are returned "directly" by the Android Touch API; examples include: the number of fragments the scroll down gesture is composed of (*fragments number*), the duration of the scroll down gesture (*duration*), the coordinates of the initial point of the scroll down gesture (X_s, Y_s), and so on. Other features regard the "geometric" properties of the scroll down gesture, and have to be specifically calculated. As an example, let $\big(X_s, Y_s\big)$ and $\big(X_e, Y_e\big)$ the coordinates of the start point and the end point of a scroll down gesture respectively, then the *length* of the scroll down gesture is defined as:

$$length = \sqrt{(X_e - X_s)^2 + (Y_e - Y_s)^2} \tag{6.1}$$

Also, let X_{max} and X_{min} (resp. Y_{max} e Y_{min}) be the maximum

Table 6.1: Description of the raw data captured through the *AI4C app* for each touch-based gesture and for each micro-game.

Game	**Gesture**	**Information**
Reading	Scroll	*start time, start point, end point, duration, touch dimension, touch pressure, velocity along x and y axis*
Pacman	Swipe	*start time, start point, end point, duration, touch dimension, touch pressure, velocity along axis*
Color Matching	Tap	*start time, start point, end point, touch dimension, touch pressure*
Score a goal	Drag & Drop	*start time, start point, end point, duration, touch dimension, touch pressure, velocity along x and y axis*
Writing	Keystroke	*time, number of characters, number of deletions*
Calculation	Pinch to zoom	*start time, finger1 start point, finger1 start dimension, finger1 start pressure, finger2 start point, finger2 start dimension, finger2 start pressure, length, fingers motion points, fingers motion velocity along x and y axis*

Table 6.2: Number of occurrences for each touch-based gesture with information about the category and the calculated features.

Gesture/Dataset	**Size**	**#Underages**	**#Adults**	**#Features**
Scroll down	2594	807	1787	50
Swipe right	972	306	666	13
Swipe left	1005	315	690	13
Tap	2942	921	2021	29
Drag & Drop	735	230	505	13
Writing	645	166	479	5
Pinch-to-zoom	1090	319	771	48

and the minimum coordinates on the x axis (resp. coordinates on the y axis) of the scroll down gesture respectively, then the covered

Table 6.3: Details about the features calculated for each touch-based gesture.

Gesture	**Features**
Scroll down	*fragments number, duration, start point (X_s, Y_s), end point (X_e, Y_e), length, total velocity, area, turning point (X_{tp}, Y_{tp}), turning point angle, acceleration$_{s,tp}$, acceleration$_{tp,e}$, start dimension, middle dimension, end dimension, min dimension, max dimension, mean dimension, 25% dimension, 50% dimension, 75% dimension, start pressure, middle pressure, end pressure, min pressure, max pressure, mean pressure, 25% pressure, 50% pressure, 75% pressure, start velocity X, middle velocity X, end velocity X, min velocity X, max velocity X, mean velocity X, 25% velocity X, 50% velocity X, 75% velocity X, start velocity Y, middle velocity Y, end velocity Y, min velocity Y, max velocity Y, mean velocity Y, 25% velocity Y, 50% velocity Y, 75% velocity Y*
Swipe right	*duration, start point (X_s, Y_s), end point (X_e, Y_e), dimension, pressure, length, velocity, acceleration, area, X velocity, Y velocity*
Swipe left	*duration, start point (X_s, Y_s), end point (X_e, Y_e), dimension, pressure, length, velocity, acceleration, area, X velocity, Y velocity*
Tap	*fragments number, X shift, Y shift, start dimension, middle dimension, end dimension, min dimension, max dimension, mean dimension, 25% dimension, 50% dimension, 75% dimension, start pressure, middle pressure, end pressure, min pressure, max pressure, mean pressure, 25% pressure, 50% pressure, 75% pressure*
Drag & Drop	*duration, start point (X_s, Y_s), end point (X_e, Y_e), dimension, pressure, length, velocity, acceleration, area, X velocity, Y velocity*
Writing	*time, number of characters, frequency, number of deletions, jaccard similarity*
Pinch-to-zoom	*finger1 start point (X_1, Y_1), finger1 dimension, finger1 pression, finger1 start point (X_2, Y_2), finger2 dimension, finger2 pression, fragments number, length, area, pinch grade, start dimension, middle dimension, end dimension, min dimension, max dimension, mean dimension, 25% dimension, 50% dimension, 75% dimension, start pressure, middle pressure, end pressure, min pressure, max pressure, mean pressure, 25% pressure, 50% pressure, 75% pressure, start velocity X, middle velocity X, end velocity X, min velocity X, max velocity X, mean velocity X, 25% velocity X, 50% velocity X, 75% velocity X, start velocity Y, middle velocity Y, end velocity Y, min velocity Y, max velocity Y, mean velocity Y, 25% velocity Y, 50% velocity Y, 75% velocity Y*

area is defined as:

$$area = \left(X_{max} - X_{min}\right) * \left(Y_{max} - Y_{min}\right) \qquad (6.2)$$

Other types of features include information about the velocity, touch dimension, and touch pressure. Specifically, for each of this information we considered: *(i)* the value with respect to the whole gesture running, *(ii)* the maximum, minimum, mean value, and *(iii)* the values in correspondence of the *quartiles* of the gesture (the value at start, 25%, 50%, 75% and at end of the gesture running).

Finally, the last features regard the *turning points*. Given a scroll down gesture, the turning point $\left(X_{tp}, Y_{tp}\right)$ is the point where the gesture changes direction respect the x axis. The information about the acceleration, is related to the turning point, and is captured by two features, i.e., the acceleration of the scroll down gesture from $\left(X_s, Y_s\right)$ to $\left(X_{tp}, Y_{tp}\right)$ and the acceleration of the scroll down gesture from $\left(X_{tp}, Y_{tp}\right)$ to $\left(X_e, Y_e\right)$. Likewise, we included information about the touch dimension and the touch pressure (see Fig. 6.5) in correspondence of $\left(X_{tp}, Y_{tp}\right)$ (indicated as *middle dimension* and *middle pressure* in Table 6.3).

Swipe data set. As shown in Table 6.2, we collected 972 swipe right gestures (resp. 1005 swipe left gestures), of which 31% belongs to underages and 69% to adults (resp. 315 belong to underages and 690 to adults). To build this data set (resp. *Swipe left data set*), for each swipe gesture we calculated 13 features, shown in Table 6.3. Specifically, we considered: duration, coordinates of the start point, coordinates of the end point, dimension, pressure, velocity along x axis, velocity along y axis, length, acceleration and, finally, the area.

Tap data set. We collected 2942 tap gestures, of which 921 belongs to underages and 2021 to adults. To build this data set, for each tap gesture we have calculated 29 features (see Table 6.3). Specifically, we have considered the start point (X_s, Y_s) and the end point (X_e, Y_e). As seen for the scroll down dataset, we have considered the number of fragments of the tap gesture, and

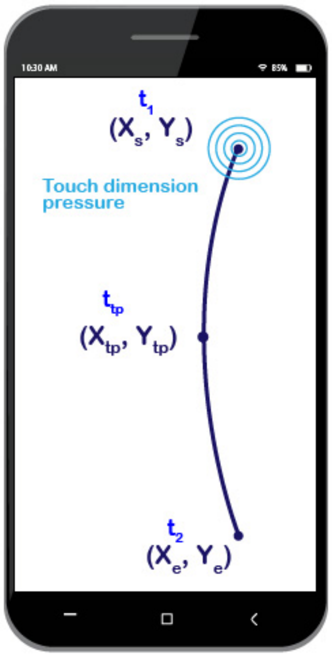


Figure 6.5: Scroll down gesture: turning point and touch information.

both for velocity, dimension and pressure, we calculated maximum, minimum, mean values, values in correspondence of the quartiles, and maximum shift respect the x axis (resp. y axis).

Drag & Drop data set. We collected 735 drag & drop gestures, of which 230 belongs to underages and 505 to adults. To build this data set, for each gesture we calculated the same features calculated for the swipe gesture (see Table 6.3).

Writing data set. We collected 645 writing gestures, where 166 belongs to underages and 479 to adults. To build this dataset, for each keystroke we calculated 5 features (see Table 6.3): time, number of characters, number of deletions, writing frequency of $\overline{S}$ and the Jaccard similarity [433] between S and $\overline{S}$, where S is the sequence of "characters to write" (proposed by the micro-game), and $\overline{S}$ is the sequence of "characters written" by the user performing the test.

Pinch to zoom data set. We collected 1090 pinch-to-zoom gestures, of which 771 belongs to underages and 48 to adults. To build this data set, for each pinch-to-zoom gesture we calculated 48 features (see Table 6.3). Among these, the finger1 start point (X_1, Y_1), the finger2 start point (X_2, Y_2), the finger1 (resp. finger2) dimension, pression, length, covered area and pinch grade. As seen for the scroll down dataset, we have considered the number of fragments of the pinch-to-zoom gesture, and both for velocity, dimension and pressure, we calculated maximum, minimum, mean values and values in correspondence of the quartiles.

6.4.3 Phase 3: ML methods

In this section, first we provide some preliminary statistical information obtained by analyzing the data sets and the features calculated in the previous phase, and then we describe the ML-based methods we proposed.

Some preliminary statistical observations

By analyzing features about pressure, dimension, duration and length, some differences between adults and underages have been found. As an example, to perform a scroll gesture, the underages required a 12% longer duration respect to the adults. Furthermore, the length of gestures performed by underages was, in pixel, 4% greater than that of the adults. The dimension of the adults' touch dimension, instead, is 23% greater than that of the underages.

Overall, by observing the features: *(i)* for each type of gesture the dimension of the adults' touch dimension is about 20% larger than that of the underages, *(ii)* the duration of swipe and drag & drop gestures of underages is greater than that of adults, *(iii)* the frequency of writing of the adults is about 116% higher than that of the underages, and *(iv)* the pressure of the touch is almost similar.

Preprocessing and validation

The aim of this phase is to prepare the data set for the validation and testing phases. The classical approach in literature for problems of touch-based gesture classification is the *single-view learning*, in which each item of the data sets contains the features associated to one single gesture [421, 420, 426]. Other approaches aim to find the best *combination* of touch gestures which can be used to classify underages and adults. Such combinations are based on the *multi-view learning* techniques [388, 434, 435], in particular *early*, *intermediate* and *late integration*. The "early integration" consists in concatenating the features associated to different gestures (single-views) performed by the same individual; in this way each combination (concatenation of two or more single-view features-vector sample) represents one sample in the data sets; this approach has the downside of considering large space features vectors. The "intermediate integration" consists in performing a features selection for each type of gesture [388, 434] (single-view), and then by combining the features selected; thus, for each individual, we combine (concatenate) such features in order to obtain the samples for the integrated datasets; the advantages of such a technique are: *(a)* the heterogeneous nature of the gestures' features can be better used by separating the data, *(b)* the size of the output is reduced, and *(c)* the separate extraction of features for different type of gestures implements the *divide-et-impera* principle, reducing the complexity of the operations. With the "late integration" we train a classifier for each type of gesture (single-view) and then we use the outputs obtained by these models as input for a new model used for the final classification [436]. This method has the advantage to be easily implemented in parallel, because each model is fitted on a single view in an independent fashion but, as downside, it does not account interactions that could exist among single views.

In this phase we tested both the single-view learning techniques and the multi-view learning techniques for combinations of pairs, triples and quadruples of gestures (Section 6.4.3). For each of these models, the data set built was split into: *(i) the training set*, obtained by including the 80% of the elements (randomly chosen), and *(ii) the testing set*, obtained including the remaining 20% of the elements. Due to the different size of the adults and underages groups, we applied to the training set the SMOTE algorithm [437], applied in several studies [438, 439, 440], to do over-sampling on the data of the smaller group. In this way, for each learning technique (single-view and multi-view), we obtained balanced datasets.

In this work we used the most popular ML models available in literature and implemented by the *scikit-learn* Python library [441], that is, Random Forest (RF) [70], Support Vector Machines (SVM) [65], MultiLayer Perceptron (MLP) [63], and Logistic Regression (LR) [442]. Finally, in order to validate the ML models we perform a 10-fold cross-validation by using the *GridSearchCV* method, as proposed in [409, 428]. The performance of the classifiers have been evaluated with popular metrics: AUC - Area Under the Receiver Operating Characteristic (ROC) curve, and accuracy [443, 444, 445]. ROC curve, the most common way evaluate the performance of a binary classifier (also used in [428]), is created by plotting True Positive Rate against False Positive Rate. The ROC AUC value ranges from 0 to 1, where 1 is the perfect accuracy score.

For each model, we evaluated the following hyper-parameters (shown in italic):

- Random Forest (RF): its performance rely mainly on the *number of estimators*, therefore we tested from 20 to 200 estimators. Best results were found between 100 and 200 estimators.

- Support Vector Machines (SVM): it was tested on different *kernels* (polynomial, sigmoid, radial) and optimized with respect the *penalty parameter C* (from 0.1 to 100). Best results were found with radial kernel and C from 1 to 100.

- MultiLayer Perceptron (MLP): it mainly relies on *hidden layers size*. The number of hidden layers size was tested from 5 to the size of input layer (according to the touch gesture considered). Best results were found between $\frac{1}{2}$ and $\frac{4}{5}$ of input layer size. We also adopted the *lbfgs optimizer* in the family of quasi-Newton methods proved to converge faster and perform better on small datasets [373].

- Logistic Regression (LR): as *optimization algorithm* we used *liblinear* (good for small datasets), and *lbfgs*, and the *penalty norms l1* and *l2*. In addition, as for SVM, we tested C parameter from 1 to 100. Best results were found with liblinear, l1 penatly norm, and C between 10 and 100.

For more details on the hyper-parameters see Scikit-learn Library [441].

Results in validation show accuracy from 73% for tap to 92% for scroll down, and ROC AUC from 0.74 for writing to 0.98 for scroll down, when using the best performing classifier i.e., Random Forest.

Classification

Here we provide details about the results of the application of different ML methods, taking into account different classifiers and combining different types of gestures. We analyzed both single-view and multi-view learning techniques. We want to emphasize that in the following we will indicate with *scrollD*, *swipeL*, *swipeR*, *tap*, *dad*, *writing* and *pinch*, the Scroll down data set, the Swipe Left data set, the Swipe Right data set, the Tap data set, the Drag & Drop data set, the Writing data set and the Pinch to zoom data set, respectively.

Single-view. In this model each sample in the data sets contains the features associated to a single touch gesture. Table 6.4 shows the ROC AUC and the accuracy score for each classifier on the test set. Best results have been obtained with the Random Forest classifier and when using the scroll down gesture, with ROC AUC of 0.93 and accuracy of 86%. For the other data sets ROC AUC values range from 0.76 to 0.83 (with Random Forest).

Table 6.4: Single-view: accuracy (*acc*) and ROC AUC (*auc*) values for each classifier and for each analyzed gesture (dataset). SVM's auc is n/a because SVM is a non probabilitic classifier.

Gesture	**RF**		**MLP**		**SVM**		**LR**	
	acc.	**auc**	**acc.**	**auc**	**acc.**	**auc**	**acc.**	**auc**
scrollD	**0.86**	**0.93**	0.81	0.87	0.85	n/a	0.74	0.73
swipeR	0.72	0.79	0.73	0.79	0.71	n/a	0.65	0.71
swipeL	0.71	0.79	0.68	0.71	0.71	n/a	0.66	0.69
tap	0.66	0.66	0.56	0.65	0.63	n/a	0.59	0.61
dad	0.70	0.76	0.68	0.79	0.68	n/a	0.68	0.73
writing	0.72	0.69	0.47	0.66	0.72	n/a	0.70	0.65
pinch	0.77	0.83	0.75	0.80	0.79	n/a	0.68	0.73

Multi-view. Each multi-view learning technique (early, intermediate and late integration) has been tested on gestures pairs, triples, and quadruples, respectively. In the following we show the results obtained for each of these cases. We have to emphasize that we show the results only for gestures that exhibited the best results in the previous analysis (when applying the single-view learning approach). We also remark that for the *intermediate integration* we used a Random Forest classifier for the feature selection, and for the *late integration* method the strategy is the following: in all the experiments (pairs, triples and quadruples) we used the best single classifier for each single-view, and then the outputs were conveyed as input of a final Random Forest classifier.

• *Gestures pairs.* As shown in Table 6.5, for the *early integration*, the best result is achieved when combining the scroll down and the pinch-to-zoom data sets, with the best ROC AUC score of 0.91. This result is slightly worse with respect to the single-view learning when using only the scroll down data set.

With the *intermediate integration*, for each dataset the Random Forest classifier selected the 10 most significant features. As an example, for the multi-view experiment *scrollD_pinch* (combination of the scroll down and pinch-to-zoom datasets) we have shrunk down the features-vector space from a *(50+48)*-dimension space (of the early integration) to a *(10+10)*-dimension space. In Table 6.5 we can see that the best results have been obtained for this combination of gestures (*scrollD_pinch*) with 0.91 and 83% for ROC and accuracy scores, respectively. As we can see in Table 6.5, the performance obtained with the late integration are lower than those obtained with the other two integration methods. This deterioration could be due to the nature of the integration technique, i.e., the error made on one of the two single-views spreads in the final classification. The deterioration has the same effect in all experiments, with the best result (ROC AUC score 0.86) obtained when combining scroll down and pinch-to-zoom, thus on the dataset *scrollD_pinch*.

Table 6.5: Early, Intermediate, and late integration: accuracy (*Acc*) and ROC AUC (*auc*) values for the Random Forest classifier.

	Early		***Intermediate***		***Late***	
Gesture pair	**acc.**	**auc**	**acc.**	**auc**	**acc.**	**auc**
scrollD_swipeR	0.78	0.85	0.79	0.87	0.81	0.82
scrollD_swipeL	0.79	0.86	0.76	0.86	0.79	0.81
scrollD_pinch	0.83	**0.91**	0.83	**0.91**	0.80	**0.86**
swipeR_swipeL	0.76	0.83	0.82	0.87	0.75	0.78
swipeR_pinch	0.86	0.89	0.80	0.87	0.80	0.81
swipeL_pinch	0.79	0.86	0.81	0.87	0.73	0.78

• *Gestures triples.* Similarly to the analysis about gesture pairs, for these experiments we report the best results only. The combinations that we do not show exhibited results lower than 0.85 for ROC AUC score and 83% for the accuracy score. As we can see in Table 6.6, by combining three type of gestures, the ROC AUC scores obtained with the *early integration* were always greater than 0.90, and the accuracy values are always greater than 85%. When applying the *intermediate integration*, for each dataset the Random Forest classifier selected the 10 most significant features. For instance, for the multi-view experiment *scrollD_swipeR_pinch* (combination of the scroll down, swipe right and pinch-to-zoom single-view datasets) the features-vector space has shrunk down from a *(50+13+48)*-dimension space (of the early integration) to a *(10+10+10)*-dimension space. Table 6.6 shows the results obtained with the intermediate integration, with the best ROC AUC score of 0.92 and accuracy of 88%, obtained when combining the scroll down, the swipe left and the pinch-to-zoom datasets. Fig. 6.6 shows the 10 most relevant features for scroll down, swipe right and pinch-to-zoom datasets. As we can see, in general the most important ones regard dimension, pressure and area of the touch gesture.

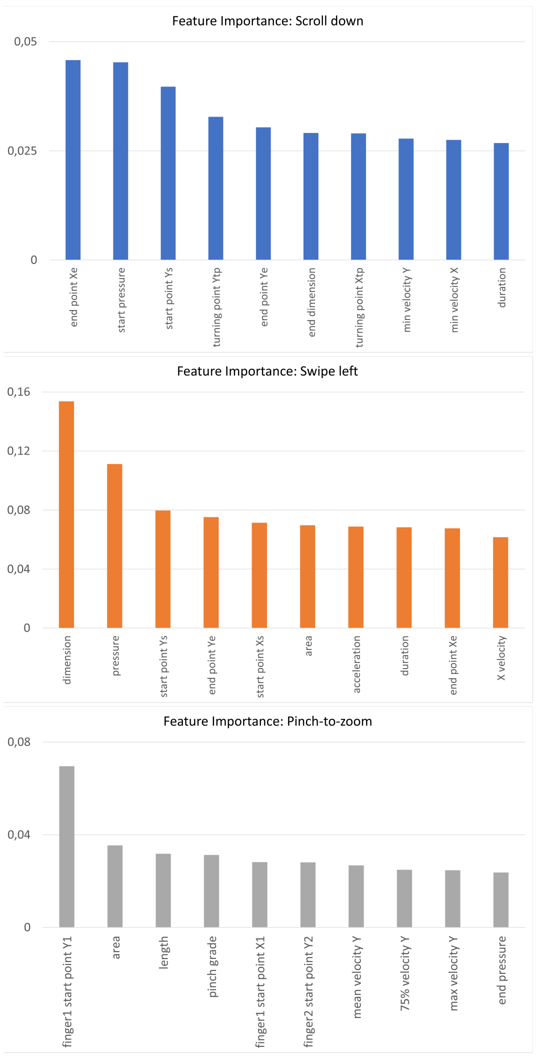


Figure 6.6: The 10 most relevant features for scroll down, swipe right and pinch-to-zoom datasets, selected by Random Forest classifier.

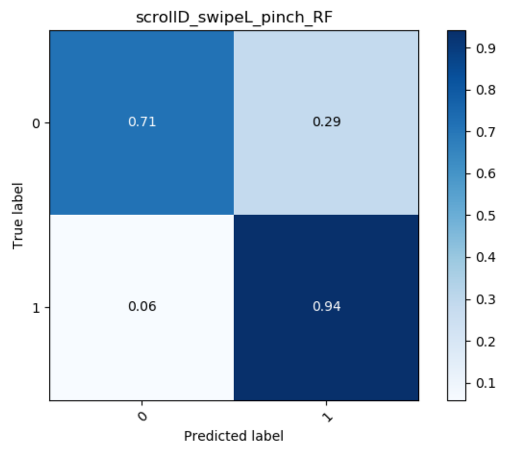


Figure 6.7: Confusion matrix for gestures triples experiment, including scroll down, swipe left, and pinch-to-zoom single-views, when applying intermediate integration technique.

Fig. 6.7 shows the normalized confusion matrix with true positive, false positive, true negative and false negative obtained results. In terms of ROC AUC scores (probabilistic measure) the classifier is very accurate (0.92), while in terms of accuracy (discrete measure) the classifier shows slightly lower performance when classifying underages (0.71). The problem is that errors (0.29) occur when classifying individuals with age near to the 16-years-old threshold. We also performed experiments in which samples did not included such individuals, obtaining results comparable to [428, 422]. This result suggest to further investigate the classification of individuals in the middle of puberty. Regarding the *late integration* (see in Table 6.6), the ROC AUC scores obtained are similar each other, in particular around 0.90. Compared to the results obtained in the experiments involving late integration and gestures pairs, by adding another single-view (another touch gesture) we obtain the same results. Whereas the late integration technique applied to gestures triples, compared with the other integration techniques, shows slightly lower performances. This indicate a correlation between single-views.

Table 6.6: Early, Intermediate, and Late integration: accuracy (*acc*) and roc auc (*auc*) values for RF classifier and for each triple of gesture datasets.

Gesture triple	***Early***		***Intermediate***		***Late***	
	acc.	**auc**	**acc.**	**auc**	**acc.**	**auc**
scrollD_swipeR_pinch	0.85	0.93	0.85	0.90	0.86	0.90
scrollD_swipeL_pinch	0.86	0.92	**0.88**	**0.92**	0.86	0.89

• *Gestures quadruples.* The last experiment considers a broader touch gestures combination. In our approach, the single-view used for the experiment has been chosen among the ones which showed the best results in the previous combinations (pairs and triples). Specifically we integrated scroll down, swipe right, swipe left, and pinch-to-zoom. As result of the with the *early integration*, by combining these four type of gestures, the best ROC AUC score is 0.90, while the best accuracy score is 0.84. As conclusion, this result does not improve the one obtained with gestures triples. When applying the *intermediate integration*, for each dataset the Random Forest classifier selected the 10 most significant features. Thus the features-vector space has shrunk down from a *(50+13+13+48)*-dimension space (of the early integration) to a *(10+10+10+10)*-dimension space. By combining these four type of gestures, the ROC AUC score is 0.89 while the accuracy score is 83%. Such a result does not improve the one obtained when applying the early integration method. Finally, with *late integration* we obtained results comparable with the ones obtained with intermediate integration techniques applied to gestures triples, i.e., 0.89 as ROC AUC and 88% as accuracy score.

6.4.4 Results

In this section we summarize the results obtained during our experiments. As we can notice in Fig. 6.8, in terms of ROC AUC score the single-view learning technique (*scroll_D*) shows the best result (0.93). In the multi-view learning setting, when combining three gestures with the intermediate integration (*scrollD_swipeL_pinch*), the accuracy increases up to 88% (our best result) and the ROC AUC score reaches almost the same result (0.92) as in the single-view setting. We also observe that further increasing the number of gestures to consider in the multi-view learning technique does not improve the accuracy of the methods. In Section 6.5, we will discuss the results obtained and we will provide further explanations/intuitions.

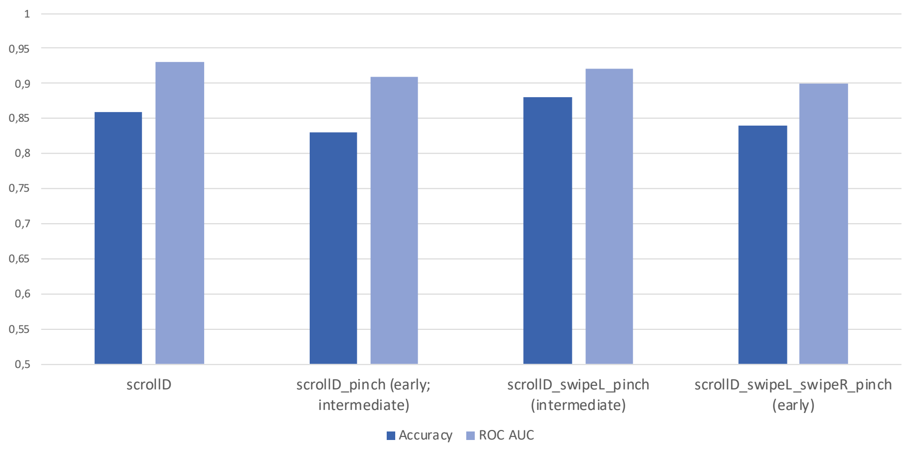


Figure 6.8: Comparison between the best single-view and multi-view learning methods proposed.

6.5 Discussion

Looking at the results, we can conclude that scroll down is the type of touch gesture that best allows us to classify among underages and adults, with a ROC AUC score of 0.93 and an accuracy score of 86%, followed by pinch-to-zoom, swipe left, and swipe right. This is largely due to the rich set of information that can be derived as features from scrolls and pinch-to-zoom. In contrast, other touch gestures are simpler; thus, only a few characteristic features can be derived. Specifically, features based on the dimension, area, and pressure of the gesture are the most informative. This shows that there is a significant variation between underages and adults in the running of touch gestures like swipes and scrolls. We remark that our data collection procedure did not impose any condition on how users needed to interact with smartphones, such as sitting, standing, or walking.

Using strategies that combine different touch gestures allows us to improve the performance of the several ML classifiers considered. Indeed, for each multi-view learning (early, intermediate, late integration) based experiment, the classifiers' average performance is better than that reached during the single-view experiments.

6.6 Conclusion

In this work, we studied touch gestures to derive whether it is possible to distinguish who is accessing a mobile device (underages or adults) to provide safeguards against threats online. Existing protection solutions are not user-friendly. Additionally, it is challenging for parents to provide continuous monitoring of children using the smartphone without impacting on their privacy and autonomy.

Therefore, our result is a new regulatory approach that exploits ML techniques to provide automatic safeguards against threats online. The idea is to relieve parents from the challenging task of configuring protecting tools and constantly monitoring children and trusting an automatic mechanism that identifies the user accessing a smartphone and provides the right protection.

The experiments have shown positive results in terms of age group classification when analyzing touch gestures on different types of smartphones. The outcome is the applicability of ML to protect children through the use of automatic approaches. As best result, the intermediate integration technique in multi-view learning method with a combination of scroll down, pinch-to-zoom, and swipe left touch gestures and Random Forest, allowed us to reach an accuracy score of 88% and a ROC AUC score of 0.92.

Given these promising results, we are currently working along two different directions. Firstly, by performing experiments with more massive and more balanced datasets to improve results and by studying other types of gestures and sequences of the same touch gesture run by the same user sequentially, as proposed in [426]. Second, in terms of techno-regulation solutions, we aim at integrating our approach inside the *AI4C app*, with a parental control system designed to support in various ways the enforcement of the safeguards resulting from the normative framework in the field of online child protection. Specifically, we are planning to extend *AI4C app* functionalities by developing different safeguards: *(i)* an awareness-enhancing software allowing to display/send alerts in case of inappropriate behaviors; *(ii)* an intelligent browser capable of filtering harmful content, according to the user accessing to the mobile device. Finally, we will perform an evaluation study in order to assess both effectiveness and user (parents) satisfaction.

Chapter 7

Expanding research scope: ML for consumer protection

Chapter Highlights

- A novel definition of Terms of Service unfairness;
- Definition of a novel method to classify the clauses within Terms of Service which obtains up to 86% F1-score in distinguishing between clauses legal typology and clauses fairness level;
- Developing of such method into *ToSware*, a Google Chrome extension making Terms of Service more readable delivering *nudge*-based protection to users so to increase their awareness;
- Real-world evaluation of *ToSware* in terms of effectiveness and efficiency that proved the efficiency of the tool and robust performance also when analyzing Terms of Service written in heterogeneous languages.
- Classification according to the taxonomy depicted in Chapter 2: **{ Type of work**: Tool, **Threat**: Terms of Service, **ML Method**: RF and SVM, **Type of protection**: Nudge **}**

7.1 Introduction

Nowadays, people use computers and mobile devices to do almost everything: to gather and share information, connect on social media, have fun, check online banking, browsing, shopping, and so on. Every app or software installed or website browsed has its own Terms of Service (ToS), i.e., legal agreements governing the relationship between providers and users, establishing mutual rights and obligations. Such contracts bind users by the time they switch on the phone or browse a website on the computer. Despite their relevance, ToS are often neglected. A recent survey exploring the behaviour shown by online users while reading ToS, reveals that consumers rarely read the contracts they accept [446].

The problem is that, whatever their content is, ToS are often too long and difficult to read [447]. It has been estimated that reading such policies alone would carry costs in time of over 200 hours/year per Internet user [448]. The cognitive effort needed and the concrete inability of laymen, i.e., users lacking technical and legal skills, of evaluating the fairness level in ToS clauses result in both a general sense of frustration for people and in making a mockery of the "notice and choice" legal regime of online ToS [449]. It is not new that some platforms make use in their ToS of unfair contractual clauses [450], i.e., "*contrary to the requirement of good faith*", causing a "*significant imbalance in the parties rights and obligations arising under the contract, to the detriment of the consumer*"[1]. Very often, they disregard not only consumer protection law but also what can be considered the EU's "*acquis*", i.e., the set of norms and principles emerging from the body of regulations binding on all EU countries. Informed consent, understood as "freely given, specific, informed and unambiguous agreement expressed through clear statements"[2] represents, in this regard, a guiding principle.

In this perspective, a relevant issue is that public agencies in

[1]See Council Directive 93/13/EEC on Unfair Terms in Consumer Contracts, article 3.1.

[2]See article 4 (11) Regulation (EU) 2016/679

charge of control concretely lack the resources needed to effectively fight against such unlawful practices. Likewise, users, researchers, and regulators still lack usable and scalable tools to cope with ToS hidden threats. Therefore, a novel solution to increase users awareness about some unfair behaviors and uphold legal safeguards becomes needful [16]. A high-level overview of the proposed system is available in Figure 7.1.

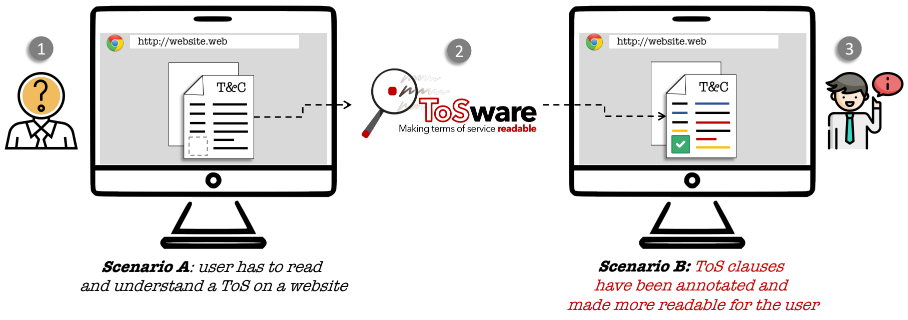

1
http://website.web
T&C
2
ToSware
Making terms of service readable
http://website.web
T&C
3
Scenario A: user has to read and understand a ToS on a website
Scenario B: ToS clauses have been annotated and made more readable for the user

Figure 7.1: An overview of the proposed system. (1) The user is concerned and unaware of the ToS content. (2) It uses our system, *ToSware*, to analyze the ToS (or part thereof). (3) ToS annotates unlawful clauses and raise user's awareness.

The main contributions of this work can be summarized as follows.

- We propose a novel definition of *ToS unfairness*. In support of such a definition we also define a novel *unfairness measure*, computed counting the unfair and potentially unfair clauses contained in a ToS, weighted via an *ad hoc weighting function* which assigns more significance to the clauses that have a direct impact on the customers concrete interests.

- We propose a novel ML-based approach to classify clauses in ToS, represented by using sentence embedding, into both *categories* and *fairness classes* (a legally determined concept that is much more complex but also growingly relevant in data mining research [451]). Support Vector Machine (for clauses' categories) and Random Forest (for clauses' fairness levels), resulted to be the most suitable methods for our specific problem after a comparing phase with other widely adopted classifiers [452, 453]. As a result, we obtained a F1-score of 86% in classifying the clauses into (a predefined set of) categories and up to 81% in classifying them according their level of fairness, i.e., potentially unfair and fair clauses. We remark that this represents an evaluation of the capabilities of widely used ML techniques to be used for classifying clauses in ToS. The nature of the problem and available dataset led us to hypothesize that this problem could be faced with basic ML techniques, without the use of complex and expensive techniques. Indeed, the obtained results confirmed this initial hypothesis.

- We compared the performance of our approach with state-of-the-art methods; results showed that approach was able to outperform all competitors with regard to for all the analyzed scores, i.e., F1-score, Precision and Recall scores.

- We embedded the proposed approach into *ToSware*, a prototype of a Google Chrome extension aiming at offer end users with increased knowledge of the categorization of ToS clauses and an increasingly awareness about their unfairness level. The tool has undergone a preliminary evaluation study to assess effectiveness, efficiency and, finally, overall users satisfaction when interacting with it.

The rest of the section is organized as follows: in Section 7.2 we discuss the main contributions available in literature by highlighting the key differences with our work. In Section 7.3 we present the rationale of our work and basic concepts useful to understand our solution. Section 7.4 is devoted to explain the problem formulation. Section 7.5 details the approach adopted to classify ToS clauses according to clauses' categories and fairness levels. Section 7.6 shows *ToSware*, a new Google Chrome extension to increase users awareness about unfair behaviors hidden inside ToS content. We also discuss here results about an evaluation study aiming at assess effectiveness, efficiency and overall user satisfaction. Finally, in Section 7.7 we conclude with some final remarks and future directions.

7.2 Related work

In this section we present the few but relevant works in the field of analyzing unfairness behaviors and ambiguous content of ToS files.

In [450] authors developed a theoretical model to partly automate the process of control of clauses' fairness in online contracts. This type of automation, deployed into a software (standalone application) called uTerms, would help human lawyers make their work more effective and efficient. The proposed model focused on unfair clauses, only. Moreover, uTerms mainly relies on the use of a dictionary of human-made rules (manually created rules) constructed starting from 20 contracts (109,000 words).

There are several differences with our work. First, in identifying unfair clauses, uTerms relies on a structure-based identification mechanism. Unfair clauses will be highlighted against a perfect match with rules inside the dictionary. Newly encountered unfair clauses (with no matching words present in the dictionary) will be not triggered and highlighted. Conversely, although we too started from a dataset of manually labeled clauses, we next trained a ML method to classify clauses syntactically different (with no common words) from those contained in the training set. In addition, authors in [450] only consider unfair clauses from 5 categories: unilateral change, unilateral termination, liability, choice of law and jurisdiction. In our work, we consider a larger set of categories (see Table 7.2), by also taking into account potentially unfair clauses and fair clauses. Concerning the implementation, our approach has been deployed into a browser extension, and thus no software installation will be required. Finally, preliminary experiments assessed its efficacy, efficiency and ease of use from the final user perspective.

The most relevant work in this field has been presented in [241]. Here authors propose a ML-based method and a tool, for partially automating the detection of potentially unfair clauses. Specifically, they offer a sentence classification system able to detect full sentences or paragraphs containing potentially unlawful clauses. Various methods to analyze terms and extract features were envisioned, including TF-IDF, Bag Of Words and Set Tree Kernel. Several ML models were then compared, such as SVM, ensemble methods, Convolutional Neural Networks. As a result, they found out that an ensemble method considering all the models they compared was able to achieve the higher accuracy F1-score (around 81%), outperforming all competitors. One of the proposed approach (the most feasible in terms of computation requirements) was implemented and developed as a web app, named Claudette. The user has to paste the text to be analyzed and the system will produce an output file that highlights the sentences predicted to contain a potentially unfair clause with also information about the predicted category, among eight pre-defined different categories, this potentially unfair clause belongs to.

With regard to this work, we have both common points and differences over various aspects. First of all, the first common point is about the dataset used to train the chosen ML-based methods. Indeed, we exploited (by also extending) their annotated corpus of 50 contracts as they made it available to the community for further research on this topic. This corpus contains contracts selected among some major players in terms of users and global relevance. The second common point is about the categorization of clauses into the eight categories they identified in their work, obtained in turn by extending the categorization presented in [454]. In our work, we further extended this categorization with an extra *neutral* category, to also take into account clauses which do not represent an issue for the consumer's rights.

Concerning the differences, these are mainly about the addressed problem and the provide solution. While authors in [241] faced the problem of identifying potentially unfair clauses, by categorizing them into eight categories, we were interested in identifying belonging categories as well as fairness levels. To the best of our knowledge, this is the first work that attempts to classify ToS clauses under both points of views. To the aim of identifying categories and all types of clauses, i.e., fair, potentially unfair, and unfair clauses, we compared several ML methods, deriving that the one based on sentence embedding and SVM is able to achieve the higher F1-score (87%), outperforming all competitors in [241]. A detailed description is presented in Section 7.5. Finally, we developed and implemented our approach into *ToSware*, a Google Chrome browser extension, letting the user to stay in the same browsing context while analyzing the ToS and to use, to get information about a specific clause, a familiar system which to interact with. To make this last point clearer, we suppose *a user has just accessed to website A. The user wants to analyze the ToS displayed.* With the systems at [241, 450] he/she has to open a software/new Web page *B* (and in case type the URL), then copy and paste the ToS into the text area changing his/her context, a task that may result cumbersome. Instead, with *ToSware*, the user has only to copy and paste the ToS he/she is interested in into the text area of the popup page or, even easier, just use the context menu on right-click, visualizing the clauses classified/explained with the corresponding highlights. A final difference is that *ToSware* has been tested to assess efficiency (in terms of system performance), efficacy (in terms of identified unfair behaviors) and usability (in terms of easiness of use and general user satisfaction). A detailed description will be provided in Section 7.6.

7.3 The big lie of online ToS

To date, the definition of practical strategies to uphold legal safeguards in digital settings is a though issue. In a "hybrid" reality in which technologies and social activities melt, new challenges are raised to legal systems as traditional regulations often look unsuitable to safeguard rights.

The 'notice and choice' paradigm, i.e., the regime based on a presentation of terms followed by an action signifying acceptance of the terms themselves (typically a click on an "I agree" button, or simply the use of the website) often fails in providing users with adequate safeguards. Despite its flaws, notice and choice still represent the cornerstone in regulating users' interaction with online services (e.g., [455]). This circumstance keeps feeding what has been defined the "*biggest lie on the Internet*" [446], the lie told by users stating "*I agree to these terms and conditions*" while, as shown in a number of surveys and reports, ToS and privacy policies are overwhelming, often ambiguous and hard to follow and understand to the few consumers who venture to read them [447] and then, basically unknown.

On top of that, there is a high chance that what a person agreed upon, without reading, includes unfair clauses concerning privacy and beyond [241, 454, 450]. According to art. 3 of the Directive 93/13 on Unfair Terms in Consumer Contracts, a contractual term is *unfair* if: *(a)* it has not been individually negotiated; and *(b)* contrary to the requirement of good faith, it causes a significant imbalance in the parties' rights and obligations, to the detriment of the consumer. In [454], authors identified five categories of *potentially unfair* clauses appearing in ToS: *(i)* establishing jurisdiction for disputes in a country different than consumer's residence; *(ii)* choice of a foreign law governing the contract; *(iii)* limitation of liability; *(iv)* the provider's right to unilaterally terminate the contract/access to the service; and *(v)* the provider's right to unilaterally modify the contract/the service. In [241], such a taxonomy has been extended introducing: *(vi)* requiring a consumer to undertake arbitration before the court proceedings can commence; *(vii)* the provider retaining the right to unilaterally remove consumer content from the service, including in-app purchases; *(viii)* having a consumer accept the agreement simply by using the service, not only without reading it, but even without having to click on "I agree/I accept". In our work we further extended such a taxonomy to include in the analysis *fair* clauses, that is clauses that do not represent an issue for the consumer's rights.

It is clear that facing the problem is not trivial; the legal mechanism alone for enforcing the prohibition of unfair clauses have often failed to effectively counter this practice so far. However, several studies available in literature envisioned the possibility for users to benefit from awareness-enhancing mechanisms that help them deal with ToS and privacy policies. In this work, we dwell on the latter to raise user awareness about the understanding of ToS clauses.

7.4 ToS unfairness: definition and measurement

In this section, we propose a novel definition of *ToS unfairness*. In support of such a definition, we also define a novel *unfairness measure*, computed by taking into account all clauses contained in a ToS.

Informally, a ToS consists of a set *clauses*, where each clause can belong to one of the categories of *clauses* introduced in Section 7.3. This categorization has been introduced in [454], extended in [241] and further extended in our work to consider clauses that do not imply unlawful behaviors, and that therefore, represent fair clauses for the consumer (see also Table 7.2). It is also possible to assign to each category a significance level (weight) that expresses the clause's impact on the customer's concrete interests. To define these weights, we involved five domain experts (legal professionals, experts in data privacy and consumer rights) asking them to tag neutral clauses and assign a weight for each of the nine categories according to their expertise. Experts' criterion consists of giving more weight to the clauses having a direct impact on the customers' concrete interests while assigning less weight to the clauses that rule how and/or where the potential suffered harm should be disputed or which laws govern the contract. The re-organization of categories according to the defined weights is shown in Table 7.1.

W (weights)	C (categories)
0	*neu*
1	*a, j, law*
2	*ch, cr, ltd, ter, use*

Table 7.1: Weights (W) for each of the clauses categories (C) assigned by domain experts in the field of Law, Privacy and Consumer Rights. A detailed description of the nine categories is illustrated in Table **??**.

To measure the unfairness of a ToS, we first classify each sentence in three possible *fairness levels*, i.e., *fair*, *potentially unfair* or *unfair*, and then we compute a *quasi*-weighted sum of the unfair and potentially unfair clauses within the ToS.

Formally, let $ToS = \{s_1, s_2, \ldots, s_n\}$ be a ToS, where s_i is a clause in the contract, for $i = 1, \ldots, n$. Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of *clauses categories*, $F = \{f_1, f_2, \ldots, f_k\}$ be the set of *fairness levels*, and $W = \{w_1, w_2, \ldots, w_u\}$ be the set of *category weights*. We define a weight function $w : C \to W$ which associate to each $c_i \in C$ a weight in W. Thus, we indicate with $w(c_i)$ the weight of c_i. Now, given $s_i \in ToS$, we indicate with $c(s_i)$ the *category* of s_i, $f(s_i)$ the *fairness level* of s_i, and $w(s_i)$ the *weight* of s_i where $w(s_i) = w(c(s_i))$. We remark that $c(s_i) \in C$ and $f(s_i) \in F$.

In this work, we set $C = \{a, ch, cr, j, law, ltd, neu, ter, use\}$, where *a*, *ch*, *cr*, *j*, *law*, *ltd*, *ter*, *use*, described in Table 7.2, have been defined in [241], and *neu* has been introduced here (more details in Section 6.4.2). We set $F = \{ff, pu, uf\}$, where $ff =$ *fair*, $pu =$ *potentially unfair*, and $uf =$ *unfair*, as proposed by [241].

Thus, we set $W = \{0, 1, 2\}$ and the function w assigns weights to clauses according to the Table 7.1.

Then, given a Term of Service $ToS = \{s_1, \ldots, s_n\}$, the overall unfairness of ToS is computed as follows:

$$uf(ToS) = \frac{1}{n} \cdot \sum_{s_i \in ToS f(s_i) \in \{pu, uf\}} w(s_i)$$

A comprehensive representation of the problem is available in Fig. 7.2.

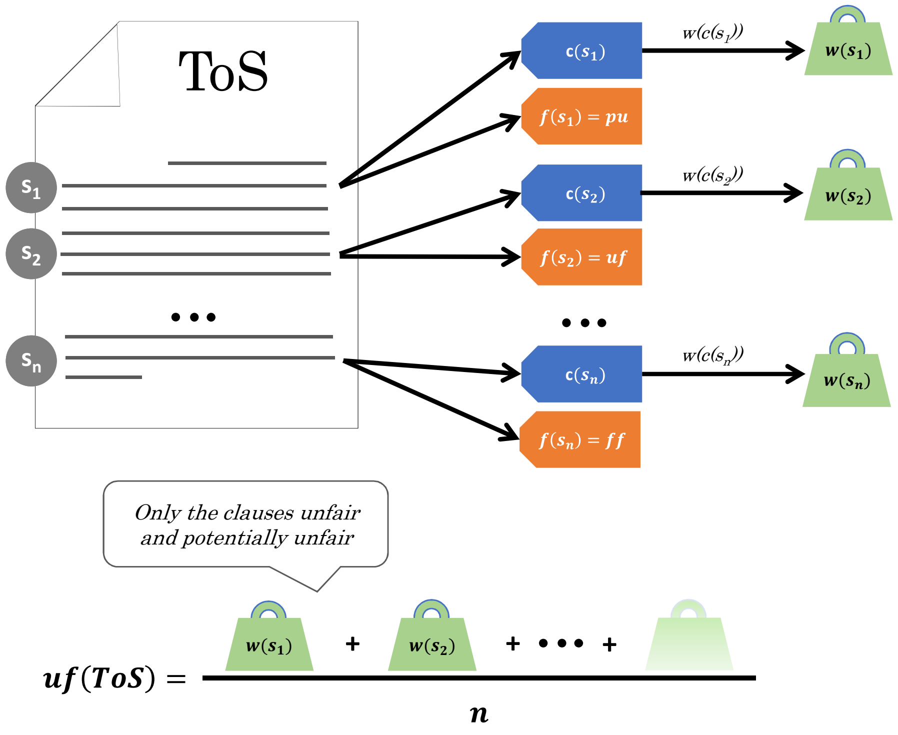


Figure 7.2: Computation of the overall ToS unfairness.

To summarize, given a ToS with a certain number of clauses to analyze, the basic idea is to: *(a)* classify these clauses according to their belonging category, *(b)* classify these clauses according to their fairness level, *(c)* assign a weight to each clause (see Table 7.1) using the result of the first classification, and finally, *(d)* compute the overall ToS unfairness (i.e., $uf(ToS)$). In Section 7.5, we will describe the methodology followed to define ML methods able to perform category classification and fairness level classification.

7.5 A ML based method to classify ToS clauses

In this section, we describe a novel ML-based method to classify ToS clauses according to pre-defined categories and a novel ML-based method to classify ToS clauses according to three fairness levels. We remark that to the best of our knowledge, this is the first work in which a ML-based method to classify ToS clauses according fairness level was proposed. To pursue this goal we defined a methodology encompassing four steps (see Fig. 7.4):

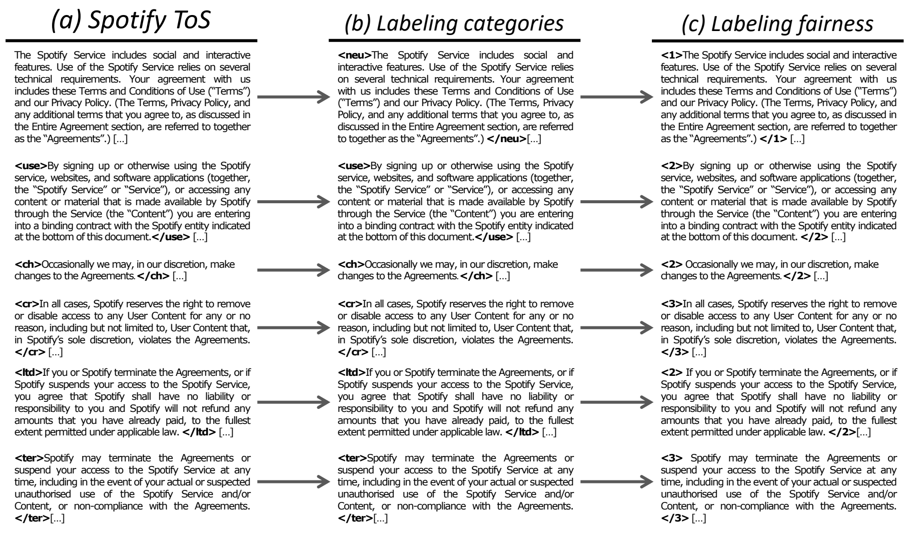
- *Dataset collection* (Section 7.5.1): in this step, we downloaded a set of `XML` formatted resources, representing the ToS files containing the clauses, labeled by authors in [241]; we updated the labeling relying on the experience of domain experts in the Law field, Privacy and Consumer Rights and we represented ToS clauses with a sentence embedding method.

- *Validation* (Section 7.5.4): the dataset has been split into training and testing sets; `k`-fold cross-validation was performed on the training set to validate different classifiers;

- *Testing* (Section 7.5.5): the most used classifiers in literature have been tested on the testing set with the best parameters found during the previous step;

- *Comparison with state-of-the-art methods* (Section 7.5.6): the most effective method, as result of the experiments did in the testing phase, has been compared with some competitors previously tested (in a earlier work) for the problem under investigation.

All experiments have been conducted with a `2,8 GHz Intel Core i7 quad-core` machine equipped with `16 GB 2133 MHz LPDDR3` RAM.



(a) Spotify ToS

The Spotify Service includes social and interactive features. Use of the Spotify Service relies on several technical requirements. Your agreement with us includes these Terms and Conditions of Use ("Terms") and our Privacy Policy. (The Terms, Privacy Policy, and any additional terms that you agree to, as discussed in the Entire Agreement section, are referred to together as the "Agreements".) [...]

<use>By signing up or otherwise using the Spotify service, websites, and software applications (together, the "Spotify Service" or "Service"), or accessing any content or material that is made available by Spotify through the Service (the "Content") you are entering into a binding contract with the Spotify entity indicated at the bottom of this document.**</use>** [...]

<ch>Occasionally we may, in our discretion, make changes to the Agreements.**</ch>** [...]

<cr>In all cases, Spotify reserves the right to remove or disable access to any User Content for any or no reason, including but not limited to, User Content that, in Spotify's sole discretion, violates the Agreements. **</cr>** [...]

<ltd>If you or Spotify terminate the Agreements, or if Spotify suspends your access to the Spotify Service, you agree that Spotify shall have no liability or responsibility to you and Spotify will not refund any amounts that you have already paid, to the fullest extent permitted under applicable law. **</ltd>** [...]

<ter>Spotify may terminate the Agreements or suspend your access to the Spotify Service at any time, including in the event of your actual or suspected unauthorised use of the Spotify Service and/or Content, or non-compliance with the Agreements. **</ter>**[...]

(b) Labeling categories

<neu>The Spotify Service includes social and interactive features. Use of the Spotify Service relies on several technical requirements. Your agreement with us includes these Terms and Conditions of Use ("Terms") and our Privacy Policy. (The Terms, Privacy Policy, and any additional terms that you agree to, as discussed in the Entire Agreement section, are referred to together as the "Agreements".) **</neu>**[...]

<use>By signing up or otherwise using the Spotify service, websites, and software applications (together, the "Spotify Service" or "Service"), or accessing any content or material that is made available by Spotify through the Service (the "Content") you are entering into a binding contract with the Spotify entity indicated at the bottom of this document.**</use>** [...]

<ch>Occasionally we may, in our discretion, make changes to the Agreements.**</ch>** [...]

<cr>In all cases, Spotify reserves the right to remove or disable access to any User Content for any or no reason, including but not limited to, User Content that, in Spotify's sole discretion, violates the Agreements. **</cr>** [...]

<ltd>If you or Spotify terminate the Agreements, or if Spotify suspends your access to the Spotify Service, you agree that Spotify shall have no liability or responsibility to you and Spotify will not refund any amounts that you have already paid, to the fullest extent permitted under applicable law. **</ltd>** [...]

<ter>Spotify may terminate the Agreements or suspend your access to the Spotify Service at any time, including in the event of your actual or suspected unauthorised use of the Spotify Service and/or Content, or non-compliance with the Agreements. **</ter>**[...]

(c) Labeling fairness

<1>The Spotify Service includes social and interactive features. Use of the Spotify Service relies on several technical requirements. Your agreement with us includes these Terms and Conditions of Use ("Terms") and our Privacy Policy. (The Terms, Privacy Policy, and any additional terms that you agree to, as discussed in the Entire Agreement section, are referred to together as the "Agreements".) **</1>** [...]

<2>By signing up or otherwise using the Spotify service, websites, and software applications (together, the "Spotify Service" or "Service"), or accessing any content or material that is made available by Spotify through the Service (the "Content") you are entering into a binding contract with the Spotify entity indicated at the bottom of this document. **</2>** [...]

<2> Occasionally we may, in our discretion, make changes to the Agreements.**</2>** [...]

<3>In all cases, Spotify reserves the right to remove or disable access to any User Content for any or no reason, including but not limited to, User Content that, in Spotify's sole discretion, violates the Agreements. **</3>** [...]

<2> If you or Spotify terminate the Agreements, or if Spotify suspends your access to the Spotify Service, you agree that Spotify shall have no liability or responsibility to you and Spotify will not refund any amounts that you have already paid, to the fullest extent permitted under applicable law. **</2>**[...]

<3> Spotify may terminate the Agreements or suspend your access to the Spotify Service at any time, including in the event of your actual or suspected unauthorised use of the Spotify Service and/or Content, or non-compliance with the Agreements. **</3>** [...]

Figure 7.3: Example of labeling performed by domain experts on the Spotify ToS. (a) The downloaded ToS file, (b) the labeling with the tag <*neu*>, (c) the labeling with the fairness level.

7.5.1 Dataset collection

The starting point of the defined methodology is the selection of ToS of online services and platforms. For this step we used ToS of popular online services made publicly available in [241]. The corpus consists of 50 online contracts, selected among those offered by some of the major players (e.g., Google, Snapchat, Spotify, Facebook, Uber, Deliveroo, Dropbox, Rovio, WhatsApp, TripAdvisor, Booking, and so on) in terms of different characteristics, such as, number of users, global relevance, and time of establishment of the service. Such files have been released in *XML* format and have been labeled by domain experts according to a categorization of the contained clauses, in the eight pre-defined categories as we described in Section 7.3.

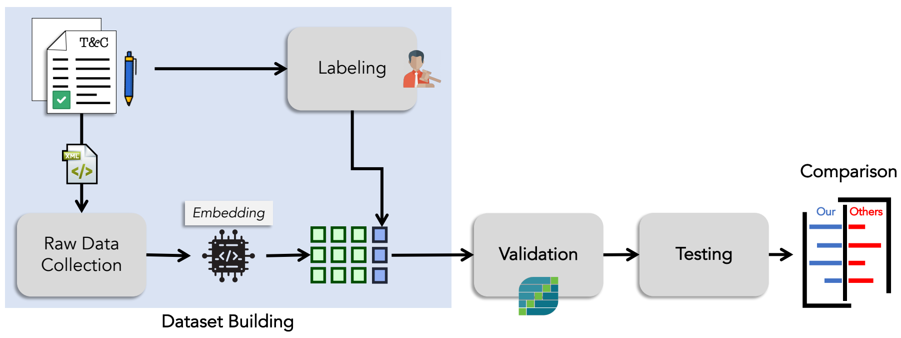


Figure 7.4: A ML-based method to classify ToS: the main steps. *Dataset Building phase*: Raw Data Collection (Downloaded ToS files), Labeling (by domain experts) and Embedding of clauses; *Validation phase*: validation of several ML methods via k-fold cross-validation and comparison of their performance; *Testing phase*: testing of the ML methods on new and unseen data; *Comparison phase*: comparison of the best method for ToS classification, identified in the Testing phase, with the ones proposed in similar works, available in literature.

7.5.2 Dataset labeling

The annotated dataset downloaded in the first phase does not take into consideration clauses extraneous to the taxonomy defined by their authors, that is, not risky clauses for consumers; as a result, these clauses were forced to fall in one of the envisioned categories despite no relevance with the target category existed (probably considering risky something that was not). For this reason, we involved five domain experts to manually annotate sentences which are âneutralâ, thus adding a <*neu*> tag to our taxonomy. The involved domain experts had a consolidated experience legal, privacy and consumer rights fields.

To better explain our procedure, shown in Fig. 7.3, we provide a clarifying example analyzing in detail the Spotify ToS. Specifically, starting from the version tagged according to the downloaded taxonomy (Fig. 7.3, *(a)*), we used the <*neu*> tag for all clauses that do not represent a risk (Fig. 7.3, *(b)*). We tagged as,

Tag	**Category Classification**	**# clauses**
<*a*>	**Arbitration.** This clause requires or allows the parties to resolve disputes through arbitration proceedings before the case can be brought to court. It is therefore considered a kind of forum selection clause.	49
<*ch*>	**Unilateral change.** This clause specifies the conditions under which the service provider may modify the terms of service and / or the service itself.	174
<*cr*>	**Content removal.** They give the provider the right to edit / delete user content, including in-app purchases, and sometimes specify the conditions under which the service provider can do it.	105
<*j*>	**Jurisdiction.** This type of clause determines which courts will have jurisdiction to judge disputes under the contract.	116
<*law*>	**Choice of law.** This clause specifies which law will govern the contract, meaning which law will be applied in a potential judgment of a dispute arising from the contract.	290
<*ltd*>	**Limitations of Liabilities.** This clause states that the obligation to pay damages is limited or excluded, for certain types of losses and under certain conditions.	221
<*ter*>	**Unilateral termination.** This clause gives the supplier the right to suspend and / or terminate the service and / or contract, and sometimes specifies the circumstances under which the supplier claims to have the right to do so.	74
<*use*>	**Contract by using.** This clause establishes that the consumer is bound by the terms of use of a specific service, simply by using the service, without even being obliged to mark that he has read and accepted them.	73
<*neu*>	**Neutral**. Sentences not falling within the taxonomy defined by [241] which do not represent an issue for the consumer's rights. All sentences of this type are fair.	100

Tag	**Fairness Classification**	**# clauses**
1	Additional tag information* for **Fair clauses**	147
2	Additional tag information for **Potentially unfair clauses**	843
3	Additional tag information for **Unfair clauses**	212

Table 7.2: Classification of categories and fairness levels: tag, description and total number of clauses. The <*neu*> tag has been added to consider neutral (fair) sentences.

as an example, <*neu*> the sentence declaring that: *The Spotify Service includes social and interactive features. Use of the Spotify Service relies on several technical requirements. Your agreement with us includes these Terms and Conditions of Use ("Terms") and our Privacy Policy. (The Terms, Privacy Policy, and any ad-*

ditional terms that you agree to, as discussed in the Entire Agreement section, are referred to together as the "Agreements".) It is obviously a sentence that is not adequate for any of the categories defined in the original downloaded taxonomy, which we rely on in our study. In this work, we added a total of 100 <*neu*> tagged sentences, obviously all clauses fair and not risky for consumers. Finally, we tagged each clause with a degree of fairness across three possible tags, and that is, <*3*> for unfair clauses, <*2*> for potentially unfair clauses and <*1*> for fair clauses (Fig. 7.3 *(c)*).

In summary, we built two datasets, the first one named *Tags*, with clauses split by categories, and the second one named *Fairness*, with clauses split by fairness levels, whose labeling information are summarized in Table 7.2. The final goal was, from one hand, to classify ToS clauses in 8+1 classes according to the *categories* (<*a*>, <*ch*>, <*cr*>, <*j*>, <*law*>, <*ltd*>, <*ter*>, <*use*>, and finally the aforementioned <*neu*> type), and on the other hand, to classify ToS clauses in 3 classes according to adequate *fairness* level (i.e., fair, potentially unfair, unfair).

7.5.3 Clauses representation

It is important to highlight that during the last years we have witnessed a flourishing of online tools enabling digital services' owners to generate, in a handful of clicks, their ToS[3]. Therefore, we can argue that is plausible a significant part of clauses can be very similar in ToS across a wide range of online services. As a consequence, the embedding of such clauses can led to a high similarity between the n-dimensional vectors, respectively. For this reason, in order to extract clauses from the raw data borrowed from [241], we employed a Python XML parser based on the ElementTree XML library. Once extracted the sentences/paragraphs, we exploited a sentence embedding method, in particular we used the multilingual Universal Sentence Encoder (mUSE) [456] (available through Tensorflow Hub[4]), to obtain one 512-dimensional vector

[3]e.g, `https://www.termsandconditionsgenerator.com`

[4]`https://bit.ly/36BSS52`

for each extracted clause (for more info, we refer the reader to Section .2). Such embedded vectors represent the features that we used for the chosen classifiers.

The assumption that the embedding of clauses in the same category can led to very similar n-dimensional vectors can be visually verified in Fig. 7.5. Specifically, for each analyzed category we have randomly chosen 8 clauses, for each clause we have calculated the 512-dimensional vector, and finally, for each pair of such vectors we calculated the similarity as their inner product. Similarities value s ranges in $[0, 1]$, where 0 means very different clauses and 0 identical clauses. As we can see, the overall similarity is above 0.4, with several areas in which the value is above 0.6.

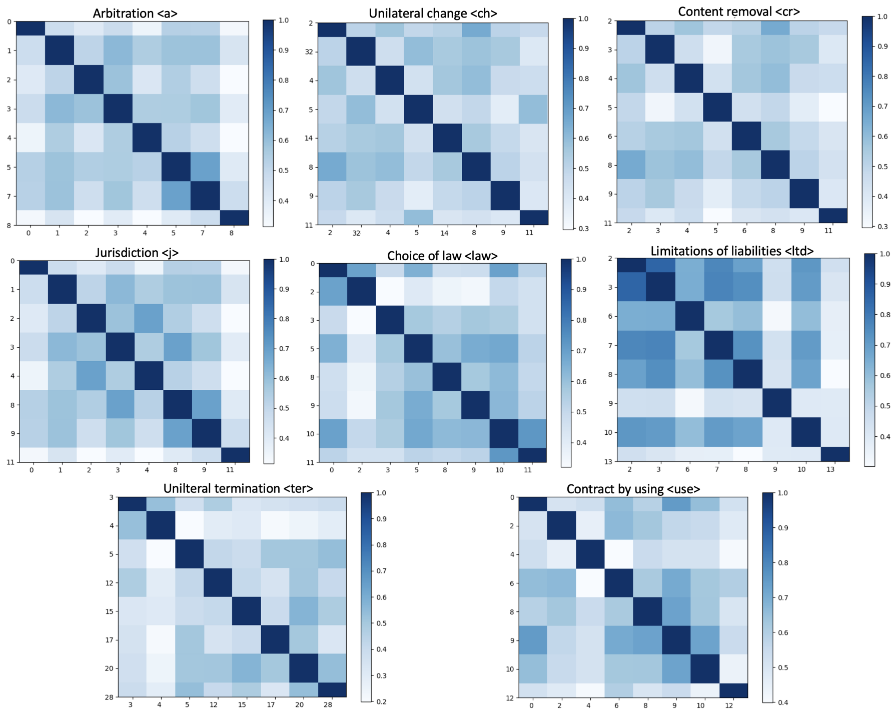


Figure 7.5: Similarities between clauses belonging to the same category.

7.5.4 Validation

The dataset built was split, with a stratified approach, into: *(1) training set*, obtained by including the 80% of the elements (randomly chosen), and *(2) testing set*, obtained by including the remaining 20% of the elements.

We compared the most popular ML methods available in literature by implementing them using the *scikit-learn* Python library, and specifically, Random Forest (RF), Support Vector Machines (SVM), MultiLayer Perceptron (MLP), K-Nearest Neighbors (KNN), AdaBoost (Ada).

Finally, to validate the ML methods we performed a 10-fold cross-validation by using the *GridSearchCV* method, as also did, as an example, in [61, 457]. In this phase, we tried to optimize the F1-score due to the slightly unbalancing in our dataset, by evaluating for each method the following hyper-parameters.

- RF: its performance mainly relies on the *number of estimators*, and therefore we proceeded with trials in the range between 100 and 1000. Best results were found between 300 and 500 estimators.

- SVM: we performed trials on different *kernels* (polynomial, sigmoid, radial) and we made optimizations with regard to the *penalty parameter C* (from 0.001 to 1000). Best results were found with the radial kernel and C values from 1 to 100.

- MLP: it mainly relies on the*hidden layers size.* The number of hidden layers size was tested with values in the range between 50 and 500 (ten by ten). Best results were found between $\frac{3}{5}$ and $\frac{4}{5}$ of the input layer size. We also adopted the *lbfgs optimizer*, that has been proved to converge faster and perform better on small datasets [373].

- KNN: We tested the *radius r* with values in the range between 1 and 5 and the *weights* parameter with *uniform* and *distance* values. Best results have been found with $r = 4$ and *uniform* weights.

- Ada: it has been tested on all the available *loss* function in the Scikit-learn library, the *number of estimators* with values in the range from 100 to 1000, and the *learning rate* with values from 0.001 to 1. Best results were found with *exponential* and *square* loss functions, *learning rate* between 0.1 and 1, *estimators* between 400 and 600.

Finally, the selected classifiers were trained with the *Tags* and the *Fairness* datasets.

7.5.5 Testing

At the end of the Validation step, we obtained the best parameters to train and test our classifiers, on the testing set. All results about both the *Tags* and the *Fairness* datasets are shown in Table 7.3. Specifically, with regard to the *Tags* dataset, the better F1-score performance, i.e., 86%, is achieved by SVM, whereas for the *Fairness* dataset, the classifier with higher performance is RF, with a F1-score of 81%. SVM and RF achieved the higher values also for Precision and Recall scores. Therefore, the outcome of this analysis is the choice to implement the classifier to distinguish the tags by using SVM and the classifier to distinguish the fairness levels by using RF. In the Scikit-learn library, for the RF classifier, there is also the possibility of implementing the classes probabilities to have as output the probabilities of labels/classes instead of the labels/classes themselves. In a real usage scenario, therefore, we can rely on the clause's probability in being fair, potentially unfair or unfair (e.g., instead of having fair/unfair as labels of clauses, we will have something like â0.70 probability a given clause is unfair").

	Classifier	***Validation***	***Testing***			
		F1 / Std. Err	**F1**	**A**	**P**	**R**
Tags	MLP	0.89 / 0.06	0.80	0.78	0.80	0.80
	RF	0.91 / 0.06	0.81	0.79	0.81	0.82
	SVM	0.93 / 0.05	**0.86**	0.85	0.86	0.87
	KNN	0.88 / 0.05	0.79	0.77	0.80	0.79
	Ada	0.91 / 0.09	0.80	0.80	0.80	0.81
Fairness	MLP	0.83 / 0.07	0.75	0.77	0.80	0.71
	RF	0.95 / 0.07	**0.81**	0.82	0.81	0.81
	SVM	0.86 / 0.08	0.76	0.78	0.81	0.71
	KNN	0.83 / 0.09	0.75	0.78	0.79	0.71
	Ada	0.92 / 0.04	0.80	0.81	0.80	0.80

Table 7.3: Classifiers' performance results for the ***Tags*** and ***Fairness*** datasets. Average F1-score achieved in the 10-fold cross-validation phase (*Validation*), and F1-score, Accuracy (A), Precision (P), Recall (R) achieved in the testing phase on the test set (*Testing*).

7.5.6 Comparison with state-of-the-art methods

The encouraging results obtained in the Testing step led us to proceed with the comparison of our approach with relevant works proposed in literature.

It is worth to note that, with regard to the category classification task, we have compared our method with those presented in [241]. Instead, as explained in Section 3 (when we analyzed differences with their work), the Fairness level classification method proposed in this work is the first attempt to classify clauses in three possible classes, i.e., fair, unfair and potentially unfair. Indeed, in [241] the authors faced a different problem, i.e., firstly how predicting whether a given sentence contains a (potentially) unfair clause and then how predicting the category to which this specific clause belongs to. Notwithstanding the lack of related works to which refer to for the efficacy of our fairness level classification method, we provide an in-depth analysis of our method's capability when addressing this task.

Results about the category classification comparison are shown in Table 7.4, and as we can see, our method, shown in the last row of the table as "SVM+mUSE" (referring to the SVM classifier used with the multilingual Universal Sentence Encoder), shows better results against all others analyzed methods. These competitors (shown in Table 7.4) represent the classifiers that authors in [241] used in their analysis, concluding with the better performance achieved by the C8 classifier, Ensemble method of C1, C2, C3, C6 and C7. We refer the reader to that paper for a detailed description of these classifiers and their combinations.

Method	**P**	**R**	**F1**
(C1) SVM-single model (sm)	0.73	0.83	0.77
(C2) SVM-combined model (cm)	0.80	0.78	0.78
(C3) Tree Kernels	0.78	0.72	0.74
(C4) Convolutional Neural Network	0.73	0.74	0.72
(C5) Long Short-Term Memory network	0.70	0.72	0.70
(C6) SVM-Hidden Markov Models sm	0.76	0.78	0.76
(C7) SVM-Hidden Markov Models cm	0.86	0.69	0.76
(C8) Ensemble of (C1, C2, C3, C6, C7)	0.83	0.80	0.81
SVM+mUSE	**0.86**	**0.86**	**0.87**

Table 7.4: Performance comparison, in terms of Precision (P), Recall (R) and F1-score, with methods available in literature [241]. We refer the reader to that paper for the description about the classifiers and their combinations.

We can now proceed with the analysis of the ability of our method to detect the clauses categories for a given fairness level. We will describe the results for each type of fairness level in turn.

Unfair clauses.

In Fig. 7.6 we show the results about the capability of our method in detecting âunfairâ clauses. Specifically, we can see results about only the *Content removal*, *Termination*, *Jurisdiction* and *Limitations of liabilities* categories, as for the specific analyzed dataset, the unfair clauses occur in such categories, only. Our method shows a very high F1-score, especially for *Jurisdiction* and *Limitations of liabilities* categories. The lower value for *Content removal* clauses can be due to the heterogeneity or less similarity between the clauses in this category. As an example if we consider: "*Rovio may manage, regulate, control, modify or eliminate virtual goods at any time, with or without notice*" (Rovio ToS) and "*You also agree that Spotify may also reclaim your username for any reason*" (Spotify ToS), their similarity is low, notwithstanding they are both *Content removal* type clauses.

Potentially unfair clauses.

Concerning potentially unfair clauses, our method has comparable performance with the best method available in literature (i.e., C8 in Table 7.4, an ensemble of different SVM classifiers). Indeed, except for *Content removal*, in which we obtained a higher precision but poor recall, all differences are negligible (see Fig. 7.7). Similarly to the previous case, this result is due to the heterogeneity of clauses within this category (see also the Fig. 7.5 where indeed *Content removal* clauses show slightly lower similarities).

Fair clauses.

In Fig. 7.8 we show results about the capability of our method in detecting fair clauses. Similarly to the analysis for discovering unfair clauses, we show here three categories as fair clauses are available only here. In addition, the best performance is obtained when dealing with *Jurisdiction* and *Limitations of liabilities* categories.

7.6 *ToSware*: a prototype tool for Terms Of Service aWAREness

Existing systems trying to make ToS easier to understand have been implemented so far as a standalone application or Web ser-

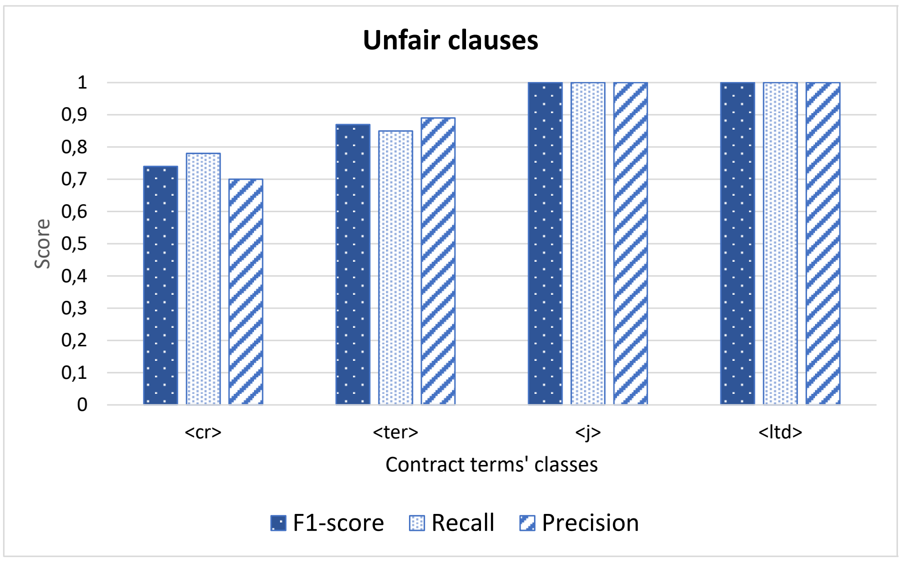


Figure 7.6: F1, Precision and Recall performance scores achieved by our method for the detection of unfair clauses.

vices. To foster the usage from any user (also without technical skills), to avoid the burden of installing specific software, and to reduce the cumbersome process of *select, copy, go to a new Web page, paste*, we embedded our approach in a Google Chrome browser extension so that the user can continue to use something familiar system (that is his/her browser), without changing context while browsing. *ToSware* is a prototype extension aiming at support individuals in evaluating ToS and better understand the (un)fairness of their clauses, just having a look at some provided information. Specifically, it provides visual aids in the form of highlighted parts, icons and probability percentages, allowing the user to assess the type/category and the "fairness-critical" level of information contained in the ToS. Awareness about the category of clauses contained in a ToS, to inform about possible ambiguous or not lawful behaviors, is guaranteed through the use of suitable and intuitive icons (see Fig. 7.10). Awareness about the "fairness-critical" level of information, to inform about the presence of un-

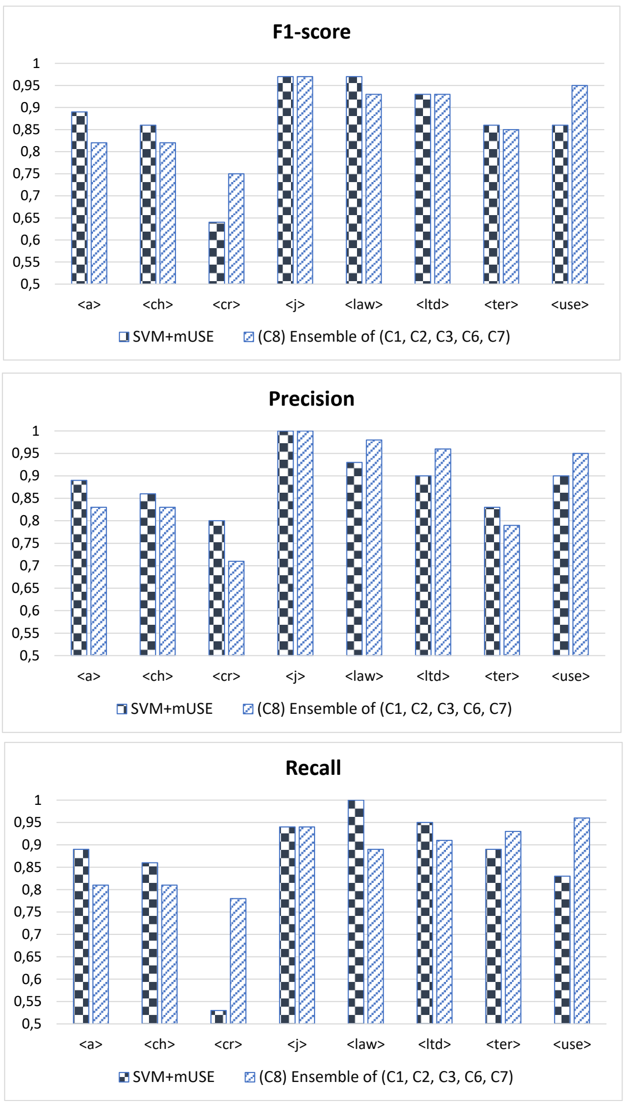


Figure 7.7: F1, Precision and Recall performance scores achieved by our method, SVM+mUSE, for the detection of potentially unfair clauses. The comparison is with the best method available in literature, an Ensemble method of 5 classifiers.

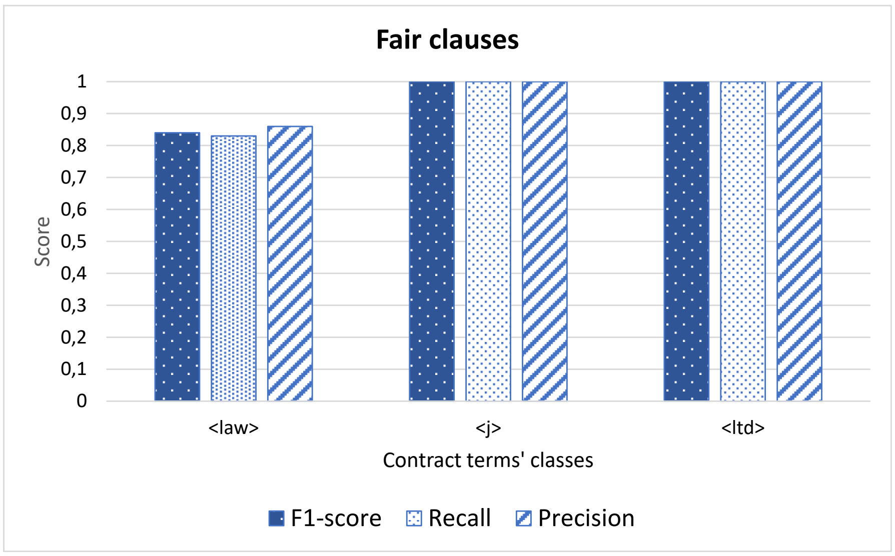


Figure 7.8: F1, Precision and Recall performance scores achieved by our method for the detection of fair clauses.

fair, potentially unfair and fair clauses, contained in a ToS file, is guaranteed through a chromatic categorization, i.e., by using red, yellow and green colors to suggest unfair, potentially unfair and fair behaviors, respectively. Finally, the probabilities inform users about the extent to which a clause can be said to have a certain level of fairness. The overall objective is to guarantee a friendly (ease-to-use interface) and effective (response-time efficiency) user experience (see Figs. 7.11 and 7.12).

The browser extension prototype[5] we designed and developed is composed, as shown in Fig. 7.9, by a client-side component and a server-side component that interact and exchange data through a request/response mechanism; we will describe these components in detail in the following, starting from a typical use case scenario. Specifically, we will first describe the *ToSware*'s architecture with the functionalities provided by its components and then the evaluation study we performed to assess efficiency, in terms of system

[5]Available at `https://bit.ly/2IWxgZ4`

performance, effectiveness, in terms of correct classification on a newly introduced dataset, and finally, usability, in terms of overall user satisfaction.

7.6.1 *ToSware* implementation

To better explain the functionalities provided by *ToSware* we describe here the typical scenario in which a user can be involved in, by giving subsequently details about the architecture and its main components.

Use case scenario

While browsing the Spotify's Terms of Service[6], the end user experiences troubles in understanding its content, and therefore he/she wishes an explanation. The steps to follow are described in the following.

1. The user inputs the ToS file or part thereof into *ToSware* UI (a simple *inputting content* action in a form, shown in Fig. **??**).

2. The text is received and elaborated by the server-side component; the ToS content is analyzed and results, that is, belonging category and fairness levels, will be returned to the client.

3. The client-side component will show the ToS clauses according to visual metaphors [348]. In particular, we make use of *customized awareness icons* for each category (see Fig. 7.10) and a simple badge for their fairness level (see Figs. 7.11 and 7.12).

[6] `https://www.spotify.com/it/legal/end-user-agreement/`

The server-side component

As shown in the rightmost part of Fig. 7.9, the server-side component in *ToSware* has been developed exploiting *Django*, a high-level Python Web framework[7]. When a ToS content is sent to the server for the analysis, *ToSware* handles it through several steps, as shown in the Algorithm 3): *(a)* detecting of the language with TextBlob[8] (line 1), *(b)* splitting the text with a well-known technique developed in [366] (line 2), *(c)* embedding the texts into vectors via mUSE (lines 3-4), *(d)* analyzing texts embedding via the *Tags Classifier* and the *Fairness Classifier* (lines 5-8). Texts and related labels are returned to the *ToSware* client-side component in JSON format. We want to emphasize that clauses classified as neutral are not returned to the client to reduce the number of response packets.

Algorithm 3: Pseudocode of tasks performed by the server-side component in *ToSware*.

```
   Input  : ToS
   Output: $clause[]$, $type[]$, $fairness[]$
1  lang ← TextBlob.langdetect(ToS);
2  $clause[]$ ← splitter(ToS,lang);
3  mUSE ← load("https://mUSEurl/");
4  $em_{clause}[]$ ← mUSE.embed($clause[]$);
5  Tags-cl ← load(SVM);
6  Fairness-cl ← load(RF);
7  $type[]$ ← Tags-cl.predict($em_{clause}[]$);
8  $fairness[]$ ← Fairness-cl.predict_proba($em_{clause}[]$);
9  return $clause[], type[], fairness[]$;
```

[7]https://www.djangoproject.com/
[8]https://textblob.readthedocs.io/en/dev/

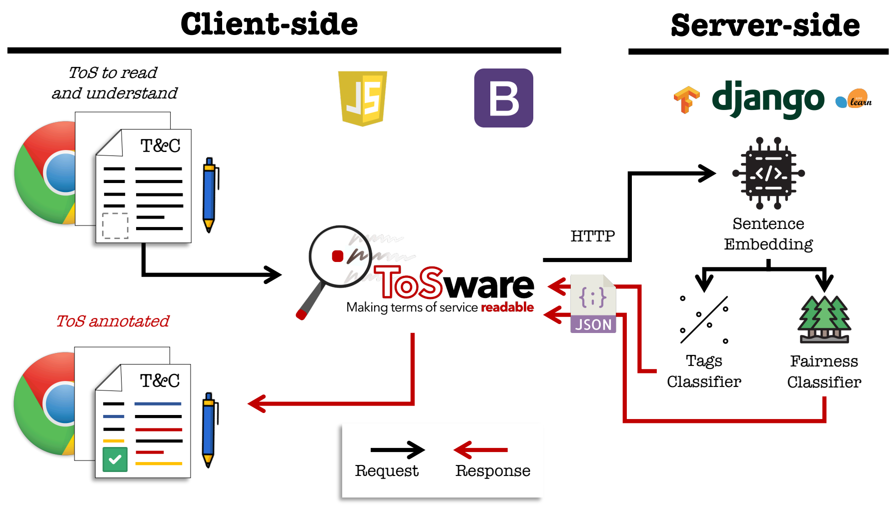


Figure 7.9: *ToSware* overall architecture and its components: the client-side component accepts users requests in terms of ToS content to understand and visualizes results highlighting them with visual clues; the server-side component handles users requests, identifies categories and fairness levels and return result to be shown to the client.

The client-side component

As shown in the leftmost part of Fig. 7.9, the client-side component allows users to input ToS content to an easy-to-use user interface implemented by using technologies such as *JavaScript* and *Bootstrap*, a popular front-end open source toolkit to quickly design and customize responsive mobile-first sites. The user has to input the ToS or part thereof into the provided form and then click on the "Analyze" button (see Fig. 7.11). As default option, we do not show clauses classified as *fair* to reduce the visual clutter; an extra action ("Show all clauses") will instead reveal them (see Fig. 7.12).

After the analysis phase performed server-side, the clauses are returned to the client-side component and are shown to the user marked with the *customized awareness icons* (Fig. ??). To ease

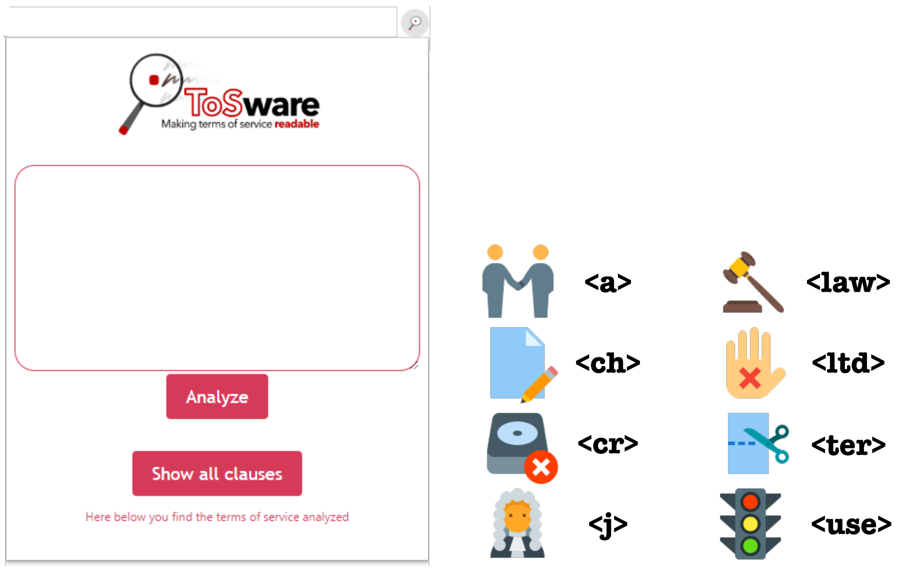
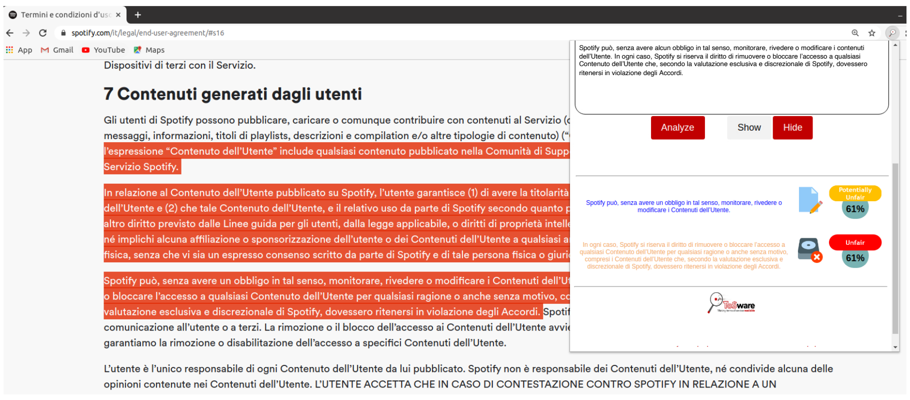

ToSware
Making terms of service readable
Analyze
Show all clauses
Here below you find the terms of service analyzed
<a>
<law>
<ch>
<ltd>
<cr>
<ter>
<j>
<use>

Figure 7.10: *ToSware* UI and visual metaphors. (a) How users can enter ToS content to analyze; (b) Icons used to make ToS clauses more readable and easy to understand for beginner users.

the understanding of this visualization we set up a complementary Web page (reachable from ToSware) summarizing the taxonomy used, and the visualization meaning in the form of a table. For each identified clause, the client also shows the fairness level through a chromatic categorization and a probability percentage that informs users about the extent to which that clause can be said to have the identified level of fairness.

7.6.2 *ToSware* evaluation

In this section we first describe the results of the browser system performance when *ToSware* loads ToS content (a new dataset of Italian ToS) and analyzes it to classify clauses and then we describe the results of the evaluation study we performed to assess the user satisfaction about the tool and its functionalities.


Termini e condizioni d'uso
spotify.com/it/legal/end-user-agreement/#s16
App
Gmail
YouTube
Maps
Analyze
Show
Hide
Potentially Unfair
Unfair
ToSware

Dispositivi di terzi con il Servizio.

7 Contenuti generati dagli utenti

Gli utenti di Spotify possono pubblicare, caricare o comunque contribuire con contenuti al Servizio (c
messaggi, informazioni, titoli di playlists, descrizioni e compilation e/o altre tipologie di contenuto) ("
l'espressione "Contenuto dell'Utente" include qualsiasi contenuto pubblicato nella Comunità di Supp
Servizio Spotify.

In relazione al Contenuto dell'Utente pubblicato su Spotify, l'utente garantisce (1) di avere la titolarità
dell'Utente e (2) che tale Contenuto dell'Utente, e il relativo uso da parte di Spotify secondo quanto p
altro diritto previsto dalle Linee guida per gli utenti, dalla legge applicabile, o diritti di proprietà intelle
né implichi alcuna affiliazione o sponsorizzazione dell'utente o dei Contenuti dell'Utente a qualsiasi a
fisica, senza che vi sia un espresso consenso scritto da parte di Spotify e di tale persona fisica o giuri

Spotify può, senza avere un obbligo in tal senso, monitorare, rivedere o modificare i Contenuti dell'Ut
o bloccare l'accesso a qualsiasi Contenuto dell'Utente per qualsiasi ragione o anche senza motivo, co
valutazione esclusiva e discrezionale di Spotify, dovessero ritenersi in violazione degli Accordi. Spotif
comunicazione all'utente o a terzi. La rimozione o il blocco dell'accesso ai Contenuti dell'Utente avvie
garantiamo la rimozione o disabilitazione dell'accesso a specifici Contenuti dell'Utente.

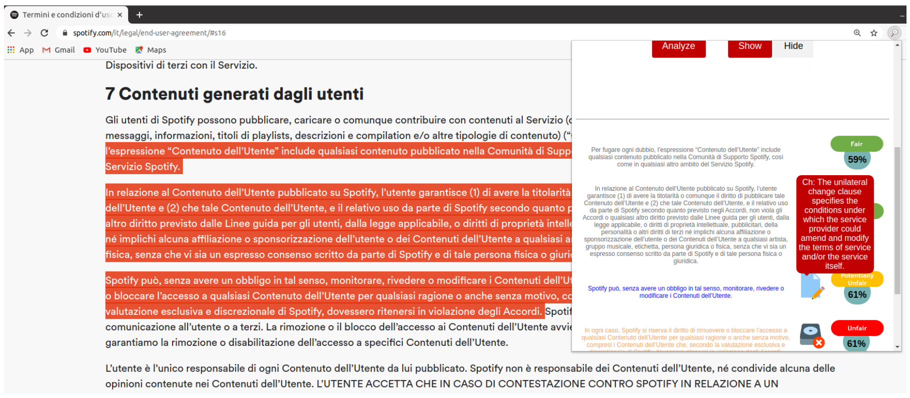
L'utente è l'unico responsabile di ogni Contenuto dell'Utente da lui pubblicato. Spotify non è responsabile dei Contenuti dell'Utente, né condivide alcuna delle
opinioni contenute nei Contenuti dell'Utente. L'UTENTE ACCETTA CHE IN CASO DI CONTESTAZIONE CONTRO SPOTIFY IN RELAZIONE A UN

Spotify può, senza avere alcun obbligo in tal senso, monitorare, rivedere o modificare i contenuti dell'Utente. In ogni caso, Spotify si riserva il diritto di rimuovere o bloccare l'accesso a qualsiasi Contenuto dell'Utente che, secondo la valutazione esclusiva e discrezionale di Spotify, dovessero ritenersi in violazione degli Accordi.

Spotify può, senza avere un obbligo in tal senso, monitorare, rivedere o modificare i Contenuti dell'Utente. — Potentially Unfair 61%

In ogni caso, Spotify si riserva il diritto di rimuovere o bloccare l'accesso a qualsiasi Contenuto dell'Utente per qualsiasi ragione o anche senza motivo, compresi i Contenuti dell'Utente che, secondo la valutazione esclusiva e discrezionale di Spotify, dovessero ritenersi in violazione degli Accordi. — Unfair 61%

Figure 7.11: *ToSware* front-end: results shown to the user asking information. Default: only potentially unfair and unfair clauses are shown to the user.


Termini e condizioni d'uso
spotify.com/it/legal/end-user-agreement/#s16
App
Gmail
YouTube
Maps
Analyze
Show
Hide
Fair
Potentially Unfair
Unfair

Dispositivi di terzi con il Servizio.

7 Contenuti generati dagli utenti

Gli utenti di Spotify possono pubblicare, caricare o comunque contribuire con contenuti al Servizio (c
messaggi, informazioni, titoli di playlists, descrizioni e compilation e/o altre tipologie di contenuto) ("
l'espressione "Contenuto dell'Utente" include qualsiasi contenuto pubblicato nella Comunità di Supp
Servizio Spotify.

In relazione al Contenuto dell'Utente pubblicato su Spotify, l'utente garantisce (1) di avere la titolarità
dell'Utente e (2) che tale Contenuto dell'Utente, e il relativo uso da parte di Spotify secondo quanto p
altro diritto previsto dalle Linee guida per gli utenti, dalla legge applicabile, o diritti di proprietà intelle
né implichi alcuna affiliazione o sponsorizzazione dell'utente o dei Contenuti dell'Utente a qualsiasi a
fisica, senza che vi sia un espresso consenso scritto da parte di Spotify e di tale persona fisica o giuri

Spotify può, senza avere un obbligo in tal senso, monitorare, rivedere o modificare i Contenuti dell'Ut
o bloccare l'accesso a qualsiasi Contenuto dell'Utente per qualsiasi ragione o anche senza motivo, co
valutazione esclusiva e discrezionale di Spotify, dovessero ritenersi in violazione degli Accordi. Spotif
comunicazione all'utente o a terzi. La rimozione o il blocco dell'accesso ai Contenuti dell'Utente avvie
garantiamo la rimozione o disabilitazione dell'accesso a specifici Contenuti dell'Utente.

L'utente è l'unico responsabile di ogni Contenuto dell'Utente da lui pubblicato. Spotify non è responsabile dei Contenuti dell'Utente, né condivide alcuna delle
opinioni contenute nei Contenuti dell'Utente. L'UTENTE ACCETTA CHE IN CASO DI CONTESTAZIONE CONTRO SPOTIFY IN RELAZIONE A UN

Per fugare ogni dubbio, l'espressione "Contenuto dell'Utente" include qualsiasi contenuto pubblicato nella Comunità di Supporto Spotify, così come in qualsiasi altro ambito del Servizio Spotify. — Fair 59%

In relazione al Contenuto dell'Utente pubblicato su Spotify, l'utente garantisce (1) di avere la titolarità o comunque il diritto di pubblicare tale Contenuto dell'Utente e (2) che tale Contenuto dell'Utente, e il relativo uso da parte di Spotify secondo quanto previsto negli Accordi, non viola gli Accordi o qualsiasi altro diritto previsto dalle Linee guida per gli utenti, dalla legge applicabile, o diritti di proprietà intellettuale, pubblicitari, della personalità o altri diritti di terzi né implichi alcuna affiliazione o sponsorizzazione dell'utente o dei Contenuti dell'Utente a qualsiasi artista, gruppo musicale, etichetta, persona giuridica o fisica, senza che vi sia un espresso consenso scritto da parte di Spotify e di tale persona fisica o giuridica.

Ch: The unilateral change clause specifies the conditions under which the service provider could amend and modify the terms of service and/or the service itself.

Spotify può, senza avere un obbligo in tal senso, monitorare, rivedere o modificare i Contenuti dell'Utente. — Potentially Unfair 61%

In ogni caso, Spotify si riserva il diritto di rimuovere o bloccare l'accesso a qualsiasi Contenuto dell'Utente per qualsiasi ragione o anche senza motivo, compresi i Contenuti dell'Utente che, secondo la valutazione esclusiva e — Unfair 61%

Figure 7.12: *ToSware* front-end: results shown to the user asking information. All clauses are shown to the user. Additionally, a tooltip can provide further explanations.

Browser performance.

For this experiment we instrumented Selenium WebDriver to perform two tasks: (1) copy-pasting into *ToSware* of 111 terms coming from 10 new Italian ToS[9], (2) click on the âAnalyze" button. These terms were previously annotated by 5 domain experts in the legal field. Meanwhile, via the psutil Python library[10] we monitored system resources usage (CPU and memory). Firstly, results showed that the time required to analyze each clause and classify it was lower than 2 seconds (the processing of 80% of requests lasted 1.3 seconds). For more 70% of the time under experimentation the RAM usage, the client-side, was under 4Mb while server-side the average usage was about 10Mb. Since the whole computation is performed server-side, the CPU usage on the client is negligible, while server-side, we had however positive results with 60% of requests that used less than 10% of the CPU.

Effectiveness.

With regards to the effectiveness of our approach on this new "previously unseen" dataset of contracts, we obtained performance F1-scores of 83% and 79% when classifying ToS clauses into categories and fairness levels, respectively. In this experiment, scores were slightly lower than those achieved in the Testing step; by analyzing in detail the sentences we found out that the majority of the errors have been made due to the different writing style between English and Italian ToS clauses (Italian clauses tend to be written in a more articulated way).

User evaluation.

For this preliminary evaluation study, we followed the standard Human-Computer Interaction (HCI) methodology [458], by envisioning three different phases, as defined and implemented in other contexts [459, 460, 461, 462]. We recruited 25 participants among computer scientist (44%) and individuals from the humanities field (56%). The sample was balanced in gender with a mean age of 33.

[9]The selected ToS, both in the original and annotated form, are available here: `https://bit.ly/2IWxgZ4`

[10]`https://psutil.readthedocs.io/en/latest/`

Prior research has shown that five users is the minimum number required for usability testing, since they are able to find approximately 80% of usability problems in an interface [463]. However, other research studies stated that five users are not sufficient and specifically, authors in [464] expressed that the appropriate number depends on the size of the project, with 7 users being optimal in small projects and 15 users being optimal in a medium-to-large project.

At a first stage, we asked participants to fill in a preliminary questionnaire asking for demographic information and information about ICT experience and skills. Then, we gave them information about *ToSware* and of its main functionalities. In the subsequent Testing phase, we asked them to use *ToSware* for a 10-minutes session and then accomplish two tasks: *(1)* "*go to* ***https://alfonsino.delivery*** *and select a single ToS clause to understand*" (Task 1) and *(2)* "*go to* ***https://www.calzedonia.com/it/*** *and select a ToS paragraph to understand*" (Task 2). At the end of each task, the users answered to questions to evaluate whether it was successfully completed, rate how easy and quick was to perform the task (standard questions from the after scenario questionnaire[11]). At the end of this testing phase we asked users to spend 10 minutes to fill in the standard Questionnaire For User Interaction Satisfaction (QUIS) [465]. Finally, in the third and last phase, the last 5 minutes were required to respond to a summary survey, in which we asked users to rate their perception about: (a) increased understanding of the proposed concepts, (b) increased awareness of the meaning of clauses, (c) the usefulness of the proposed instrument and finally, (d) their behavioral intention to use *ToSware* in the future.

Results of the evaluation study showed, firstly, that all participants rated (on a 5-point Likert scale) as very easy perform the assigned tasks (M=4.0 for both tasks). As depicted in Fig. 7.14, the analysis of the software usability through the QUIS questionnaire showed that, on average, all posed questions were rated very positively, confirming that participants were highly satisfied with

[11] https://garyperlman.com/quest/quest.cgi?form=ASQ

the software proposed.

Finally, also questions posed in the Summary survey questionnaire, and shown in Fig. 7.13, were all positively rated. Specifically, at the questions: Q1: "*Do you understand the meaning of the shown highlights*?" and Q2: "*Do you understand the meaning of the shown visual metaphors (icons)*?", most of participants provided a positive response, with only 8% of participants that not understood the use of icons (M=4.5, SD=0.6 for Q1 and M=3.9, SD=1.4 for Q2).

Moreover, 72% of participants stated that *ToSware* was able to facilitate the understanding of critical clauses (M=3.9, SD=0.7 for Q3). Finally 88% rated *ToSware* a useful instrument ((M=4.3, SD=0.7 for Q4), while 80% of participants will continue to use *ToSware* in the future (M=4.4, SD=0.8 for Q5).

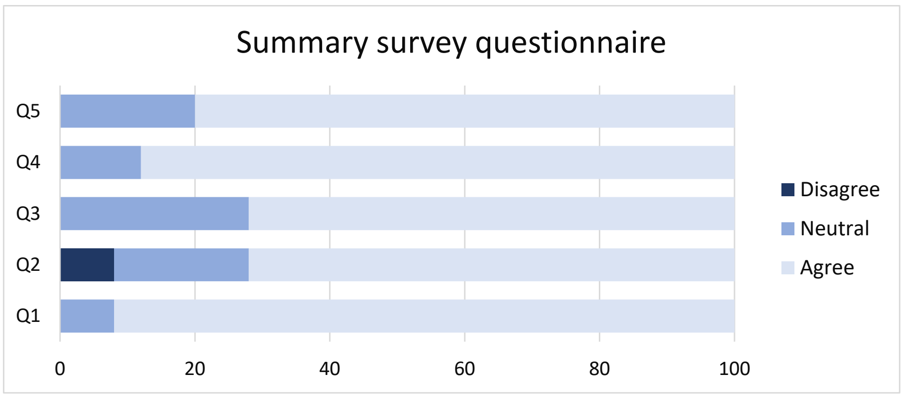


Figure 7.13: Summary survey questionnaire. Rating on a 5-point Likert scale with "Strongly agree=5" and "Strongly disagree=1" as verbal anchors. Results grouped in Agree, Neutral and Disagree classes.

7.7 Conclusion

The "notice and choice" legal regime used to rule the agreement on online ToS has shown severe flaws. Due to various reasons

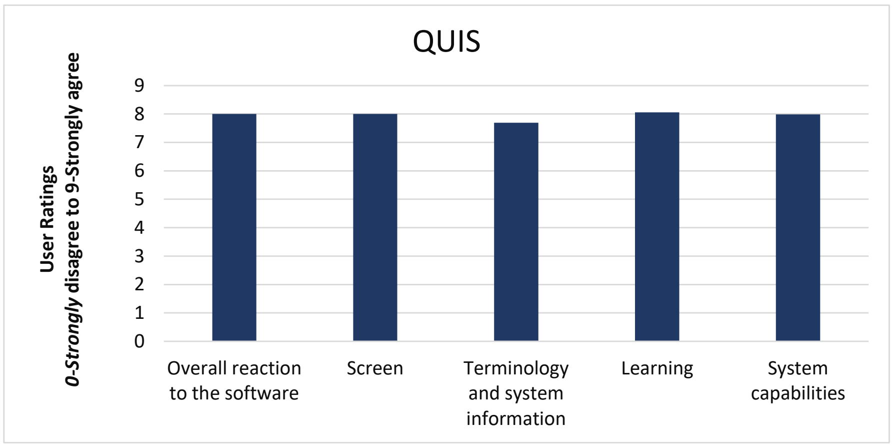


Figure 7.14: QUIS results. Rating on a 10-point Likert scale. Cronbach's $\alpha = .88$ (The internal consistency reliability among the multi-item scales [466]).

(intricacies in the texts, lack of legal skills), users struggle to grasp all the implications of clauses they are agreeing upon and often end up skipping reading them. This allows companies to take advantage of inscrutable and unfair contractual clauses that limit their liabilities or allow them to arbitrarily interrupt services at any time.

To tackle this issue, we proposed a novel ML-based approach whose main goal is to make ToS more readable/understandable in order to increase user awareness and "technologically enhance" consumer rights protection. Our approach exploited SVM for the clauses category classification task (F1-score of 86%) and RF for the fairness level categorization task of such clauses (F1-score of 81%).

We analyzed the efficacy of our approach comparing it, with regard to the clauses category classification task, with methods available in literature, outperforming them in terms of F1-score, precision and recall performance metrics. Concerning the fairness level categorization task, we did not make any comparison, since our work represents the first attempt to make this type of classification. However, when analyzing our method's efficacy, we achieved interesting results in discovering the correct fairness levels of the analyzed ToS files.

The approach has been embedded in *ToSware*, a prototype of a *Google Chrome* extension, which has been evaluated to analyze usability and impact on the user experience. The analysis of browser performance showed that the impact of the user experience is very low. The user study, involving a sample of 25 individuals, showed that all participants found out easy to use the system, and that the majority of people increased their knowledge about the unlawful practices in ToS clauses. Moreover, participants expressed high satisfaction about the tool and its usefulness, concluding with their intention to continue to use it in the future. Finally, the prototype's code is available on GitHub[12].

7.7.1 Expected impact.

The proposed approach appears to be capable of relevant impacts on both the application and research field. About the former, *ToSware* can feed innovative and socially useful applications:

- new assessment instruments of ToS fairness for public and private bodies committed to consumers rights protection in digital environments[13]. The idea is that they can use *ToSware* to analyze ToS of online services on a large scale to measure their *overall unfairness*. In this way, it could be possible to establish a threshold which once overcome makes the contract unacceptable and susceptible of further legal investigations. For example, by using our new definition of *ToS unfairness*, we obtain, for three of the ToS used in our analysis, $uf(Amazon.com) = 0.31$, $uf(Booking.com) = 0.26$, $uf(LinkedIn.com) = 0.40$. Therefore, by setting, as an example, a threshold to the value of 0.35, LinkedIn.com could be subjected to legal investigation;

[12] `https://github.com/alfonsoguarino/ToSware`
[13] `https://www.altroconsumo.it/`

- new tools to influence the behaviors of digital market operators exploiting the power of nudges [16, 467], more precisely those enabled by data-driven "socially distributed control" and reputation. As an example, *ToSware* could push the Digital Labour Platforms to improve the fairness and conditions of the employment of crowdworkers;

- new solutions that, adding user-readable meta-information to the ToS clauses, enhance the transparency of the Digital Single Market and uphold the protection of rights that, also due to âmagnitude" reasons, often transcend consumer rights protection and end up involving civil and even political issues/rights. In the same way that one can exploit mechanisms such as the priority levels defined by the Web Content Accessibility Guidelines[14] or the 5-star deployment scheme for Open Data[15], a market operator can be interested in applying meta-information to the ToS clauses to increase transparency towards its customers.

From a research point of view, *ToSware* paves the way to more in-depth explorations facing the creations of "intelligent safeguards", in line with the computational evolution of social life regulation and with most recent trends of both techno-regulation [13] and computational legal studies [468]. In particular, *ToSware* issues the new challenge of incorporating, into computational safeguards, concepts that, however "blurred" and springing from social and cultural construction processes, are crucial for regulating our societies.

7.7.2 Future works

With regards to future works, we are currently working towards three directions. First, we will define *ad hoc* models for Categories and Fairness level classifications, and we will also enlarge

[14] https://www.w3.org/WAI/
[15] https://5stardata.info/en/

the dataset of annotated ToS in order to perform further experiments with ToS in several languages. The second direction is about the employment of our strategy in the field of privacy policies; the idea is to verify whether ML based methods could be efficiently employed to identify ambiguous behaviors in policies governing the privacy of individuals and of their personal data. Finally, further design and development enhancements are planned about *ToSware*, with the final goal of making it available soon on the Google Chrome web store. Thereafter, a long lasting user evaluation and an exhaustive performance benchmarking [469, 470] will be performed to assess the overall usability and efficiency.

Chapter 8

Final remarks

In this dissertation, we faced one of the emerging issue in the digital society, that is *protecting rights*. As it happens for physical goods in physical world, protected by physical artifacts, in the digital society there is the increasing need of not only appropriate rules but also technologies directly protecting the rights. In particular, the hybrid and complex reality we live in demands for non static solutions adapting to the ever evolving number and kind of threats. In this respect, machine learning appeared to be a precious ally capable of upholding legal standards for safeguarding rights, as also emphasized by the systematic literature review conducted to build the research background.

In more details, the dissertation focuses on *privacy protection* and other related rights that can be intruded through the privacy violation, that is *child* and *consumer protection*. The research, drawing upon recent development in the areas of computational law and techno-regulation, has explored how state-of-the-art machine learning approaches can be fit to support the protection of privacy and related rights. We have faced the following *specific challenges*: *(a)* privacy protection against third-party tracking on the Web, *(b)* privacy protection against the unaware/uncontrolled dissemination of personal and private information online, *(c)* child protection on mobile devices, *(d)* consumer protection against unlawful practices in Terms of Service online.

Based upon the specific challenges, we showed how supports can be traced-back to two categories: enforcement (i.e., solutions which hamper breaches of norms) and nudge (i.e., solutions which increase user awareness to promote norm compliance).

We demonstrated that ML can be a viable support for rights protection through: *(a)* "in silico" experiments, that is in a very controlled environment, *(b)* "real-world" experiments, putting ML in practice embedding envisioned and previously tested approaches into tools for final users that have been evaluated in a more relevant environment and when possible also with final users.

Here below, we summarize research scope, threats faced, and results obtained:

a) ML for *privacy* enforcement

Threat: third-party tracking on the Web. *Results*: we have demonstrated the designed machine learning approach is capable of classifying between trackers and functional resources on the Web. We have embedded our approach in *GuardOne*, a tool to protect users' privacy, providing *enforcement* solutions to block trackers, in an effective and efficient manner.

b) ML for *privacy* awareness

Threat: unaware/uncontrolled dissemination of personal and private information online (in text format). *Results*: we have demonstrated the designed machine learning approach is capable of classifying *(a)* the text topic, *(b)* the sensitiveness of the content according to the topic. Furthermore, we have demonstrated how machine learning can be fit to learn users' attitudes towards privacy and adapt the classification of sensitive contents. We have embedded our approach in *Knoxly*, a tool to protect users' privacy, providing *nudge*-based solutions, that is warnings and alerts, to raise users' awareness against what they are about to disclose online, in an effective (when dealing with long messages) and efficient manner.

c) ML for *child protection*

Threat: children rights, illegal or inappropriate contents, conducts and contacts for children, on mobile devices. *Results*: we have demonstrated the designed machine learning approach is capable of distinguishing between underages and adults based on the analysis of a few touch gestures performed on the mobile device. The outcome can be seen as the baseline to uphold legal standards for online child protection. In fact, once identified the user, it is possible to apply the specific safeguards needed.

d) ML for *consumer protection*

Threat: Unlawful practices in online Terms of Service. *Results*: we have demonstrated the designed machine learning approach is capable of classifying the Terms of Service clauses both on the legal category and the fairness level. The envisioned solution can be used to measure the overall unfairness of online Terms of Service. We embedded our machine learning approach in *ToSware*, a tool to protect consumer by adopting *nudge*-based solution aimed at raising their awareness with respect to (potentially) unfair clauses within Terms of Service.

Overall, we have used existing methods and approaches in a novel fashion declined for different contexts and challenges. What primary emerges from the research activities is the need to go beyond, the need for *ad-hoc* solutions both on the algorithmic and regulation levels. Solutions proposed only *in silico* must be validated with real users and, before that, thought and designed in close contact with them.

In this dissertation, we have found areas and issues at the boundaries of computer science and computational law needing to be fully explored. The scientists focused on machine learning research will have to deal which such challenges because the digital age we live in demands for new specific algorithms, datasets for testing, and, more crucial, cross-fertilization between disciplines (computer science, law, education, and so on). As a matter of fact, privacy is a legal concept growing in turn with law and ICT development. Thus, computer scientists must relate to the privacy concept (and computational legal scholars) to be more compliant with the current and evolving constitutional and international laws [471] that, together with the technology advancement, always put new challenges and new issues to manage in order to better support the techno-regulation paradigm, in which *code is law*. IT researchers must also relate to privacy because of huge social planetary issues needing systemic responses (technical and legal) that, in turn, to be properly addressed, require to cross the borders between computer science and law. For instance, it could be mentioned the recent ban from Twitter of Donald Trump's profile[1]. If it is true that we witness an example of *code is law*, on the other hand Twitter has banned an individual's account which, as all other citizens, has the right to free-speech. This application of *code is law* should concern everyone when private companies like Facebook and Twitter wield the unchecked power to remove people from platforms that have become indispensable for the speech of billions.

In other words, we are facing a recursive pattern: if, on the one hand, it is real that *code is law*, it is also real that we need to "juridify" the code to uphold fundamental rights. This can be only made directly by code (again), hence the need for interdisciplinary research!

[1]https://blog.twitter.com/en$_u$$s/topics/company/2020/suspension.html$

Chapter 9

Appendix

.1 Systematic literature review: identified studies

In the following we show the most important reviewed studies describing:

1. the type of work (*Type*) which could be either "Study" or "Tool" or "Framework" (Frmwrk);
2. the *Threat* faced which could be either "Cyber Attack" (CyAtt), "Phishing" (Phish), "General data leakage" (GenLk), "Disclosure" (Dis), "Image leakage" (ImgLk), "Location leakage" (LocLk), "Speech leakage" (SpLk), "Health leakage" (HLk), "Malware" (MLW), "Privacy Policy" (PP);
3. the Highlights of the work in terms of Focus, Method, Results;
4. type of protection provided (*Prot.*) which could be either "Enforcement" (E) or "Nudge" (N) or "Not Available" (n/a) or both (N/E).

Blockchain In Table 1 we show the identified studies in the area of Blockchain.

Table 1: Identified studies in the area of Blockchain. CyAtt = Cyber Attack, E = Enforcement, N = Nudge.

Ref.	Type	Threat	Highlights	Prot.
[85]	Study	CyAtt	***Focus.*** Using deep learning to highlight anomalies in the underlying blockchain. ***Method.*** To define an anomaly detection system based on a encoder-decoder deep learning model, that is trained exploiting aggregate information extracted by monitoring blockchain activities. ***Results.*** Experiments on complete historical logs of Ethereum Classic network prove the capability of the model based on autoencoders to effectively detect the publicly reported attacks. The relevant features are: (i) the average size (in bytes) of a block; (ii) the average provided *Gas*[1] needed to perform the transaction; (iii) the average difficulty (i.e. the effort) necessary to validate a block; (iv) the average number of transactions contained in a block; (v) the total amount of employed *Gas*; (vi) the total number of transactions in all blocks.	E

Online Social Network In Table 2 we show the identified studies in the area of OSN.

Table 2: Identified studies in the area of Online Social Network. GenLk = General data leakage, Dis = Disclosure, ImgLk = Image leakage, AnUsr = anomalous users, LocLk = location leakage, CyAtt = Cyber attack, E = Enforcement, N = Nudge.

Ref.	Type	Threat	Highlights	Prot.
[185]	Study	CyAtt	***Focus.*** Authors propose a novel content-based method to detect Sybils. ***Method.*** The proposed method is an end-to-end classification model that extracts features directly from the input data, and then output the classification results in a unified framework. ***Results.*** CNN and bi-SN-LSTM can automatically extract useful features for Sybil Detection	E
[187]	Study	Dis	***Focus.*** An automated model to score the privacy of unstructured information and warn users regarding the textual data privacy risks they have shared in online social platforms. ***Method.*** Authors define a privacy score. An open dataset of social networks messages from Kaggle repository has been obtained for the training set. To extract the features, they first applied both uni-grams and POS bi-grams tagging techniques. Both Naive Bayes and J48 decision tree ML techniques are utilised to determine which scenario comes with the best F-score and precision. ***Results.*** Naive Bayes classifier shows best performance for all the conducted experiments	N
[181]	Study	ImgLk	***Focus.*** Propose a novel algorithm to protect the sensitive attributes of images against machines, meanwhile keeping the changes imperceptible to humans. ***Method.*** Authors used the EMOtional attention dataset (EMOd) as input for a CNN. ***Results.*** Authors discover that human sensitivity is influenced by multiple factors, from low-level features such as illumination, texture, to high-level attributes like object sentiment and semantics. Authors demonstrate the efficacy of the proposed model for privacy protection for images of general scenes achieving also human imperceptibility.	E
[188]	Study	LocLk	***Focus.*** Authors investigate how users can measure the privacy of their geo-location on OSN and to control the factors affecting it ***Method.*** Authors define the privacy of a target user as the geographical distance between her actual unexposed location and the location estimated by an attacker. Features cover a broad range of parameters that users can tune to control their privacy, and fall into three main categories, namely mobility-related, topology-related, and data-perturbation-related. The data-perturbation-related feature is the level of data perturbation the user is willing to adopt for tuning her privacy. ***Results.*** RF 1.4km error in estimating in a privacy preserving way the location of a user	n/a

Continued on next page

Table 2 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[177]	Study	ImgLk	***Focus.*** Authors explore the hypothesis that some generic patterns of private images can be well identified when a group of online images are taken into consideration, regardless of their authorsâ individual privacy bias and level of awareness. ***Method.*** Authors analyze privacy needs of images on a multi-level scale, consistent with current privacy options offered by most popular OSN. They adopt a five-level privacy model, where image disclosure can range from open access to disclosure to the owner only. ***Results.*** SVM performs consistently better than the other models in predicting image privacy.	N
[21]	Study	Dis	***Focus.*** Scoring Private Information in Social Networks ***Method.*** Authors present the first quantitative model for private information assessment, which generates a PrivScore that indicates the level of sensitiveness of text content. They examine usersâ opinions on the levels of sensitiveness of content, and then build a semantic model that comprehends the opinions to generate a context-free PrivScore. The model learns the sensitiveness of the content from text features (word embeddings) and sentiment features using a LSTM network ***Results.*** LSTM trained with GloVe extractred features from tweets achieves an average precision of 0.85 and an average recall of 0.85.	N
[184]	Tool	AnUsr	***Focus.*** Characterizing and Detecting Malicious Accounts in OSN. Specifically, the authors study the patterns in friend requests to distinguish malicious accounts. ***Method.*** Authors perform a systematic study over 1 million labeled data from WLink, a real social media with billions of users, to confirm their hypothesis. Based on the results, they propose dozens of new features and leverage ML to detect malicious accounts. The features can be classified into non-textual ones (e.g., number of bound bank cards, number of unique channels used to send friend requests) and textual ones (generated from the request messages). ***Results.*** The only robust account-related features are gender and the number of bound bank cards. RF achieves precision recall and f score close to 100%	N/E
[186]	Study	ImgLk	***Focus.*** In this paper, the authors propose an approach for fusing object, scene context, and image tags for predicting the privacy of images shared online. ***Method.*** The proposed approach identifies features of object, context and image tags on the fly. Authors experiments ML methods for predicting image privacy on both individual modality (only one set of features) and multiple-modalities (all sets of features). ***Results.*** Experimental results show that the proposed approach based on convolutional neural networks predicts the sensitive content more accurately than the models trained on an individual modality (object, scene, and tags).	N

Continued on next page

Table 2 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[179]	Study	Dis	***Focus.*** Privacy detection of users' unstructured data (text posted). ***Method.*** The authors present a fine-grained privacy detection network (GrHA) equipped with graph-regularized hierarchical attentive representation learning. GrHA aims to learn a latent space that is capable of characterizing the correspondence between the UGC and its labels, i.e., the personal aspects it reveals. ***Results.*** Extensive experiments on a real-world dataset well validate the proposed GrHA and demonstrate the necessity of integrating both the hierarchical attentive representation learning and graph-based semantic regularization in the context of fine-grained privacy detection. Interestingly, authors find that different words/sentences do have different confidences in revealing the usersâ privacy, and the word-level attention mechanism contributes more to the privacy detection compared to the sentence-level one.	N
[180]	Study	Dis	***Focus.*** Friend classification system in OSNs to evaluate the privacy level of a post. ***Method.*** The authors develop a supervised model to estimate the friendship strength based upon 23 different features comprising of structure based, interaction based, homophily based and sentiment based features. ***Results.*** The authors evaluate the model on a real-world Facebook dataset. The model obtains a ROC AUC of 0.82 in identifying acquaintances from all the other friends categories, and a ROC AUC of 0.85 in distinguishing between close friends and acquaintances. The experiments suggest that features like average comment length, reaction scores for likes and love, friend tag score, and Jaccard similarity consistently perform better in predicting friendship strength across different ML methods.	N
[190]	Study	Dis	***Focus.*** Predicting the privacy of hashtags. ***Method.*** They concentrate in particular on location, which is recognized as one of the key privacy concerns in the Internet era, using hashtags as features for RF method. ***Results.*** Authors show that it is possible to infer a user's precise location from hashtags with accuracy of 70% to 76%, depending on the city. They introduce a system called *Tagvisor* that suggests alternative hashtags if the user-selected ones constitute a threat to location privacy. Their findings show also that obfuscating as little as two hashtags already provides a near-optimal trade-off between privacy and utility in their dataset.	N

Mobile In Table 3 we show the identified studies in the area of Mobile.

Table 3: Identified studies in the area of mobile. GenLk = General data leakage, Dis = Disclosure, ImgLk = Image leakage, AnUsr = anomalous users, LocLk = location leakage, CyAtt = Cyber attack, E = Enforcement, MLW = malware, N = Nudge, Privacy Policy = PP .

Ref.	Type	Threat	Highlights	Prot.
[158]	Study	MLW	***Focus.*** A data-driven characterization of the principal factors that distinguish modern Android spyware both from goodware and other Android malware. ***Method.*** The authors extracted static and dynamic features from the logs obtained through the Koodous service, then used both traditional and deep ML to find the best solution for malware detection. ***Results.*** The final solution is an Ensemble Late Fusion (ELF) architecture that combines the results of multiple classifiersâ predicted probabilities to generate a final prediction.	E
[126]	Frmwrk	GenLk	***Focus.*** Proposal of LeakSemantic, a framework that can automatically find abnormal sensitive network transmissions from mobile apps. ***Method.*** LeakSemantic consists of a hybrid program analysis component and a ML component. The program analysis component combines static analysis and dynamic analysis to identify sensitive transmissions. Authors collected malicious sensitive transmission from the Android Malware Genome project. LeakSemantic extracted 1223 malicious sensitive transmissions and collected the corresponding traffic. ***Results.*** The authors use DT as classifier for abnormal network transmission detection (91% correctly recognized). Among the 1223 malicious leaking transmissions extracted from the malware dataset, they found that 69.7% of the transmissions used encryption to hide the hostnames. Malware leverages encryption to evade traditional signature-based detection approaches. To illustrate how important the decryption is, the authors conducted an experiment that trained a model based solely on unencrypted instances and tested the model on the instances with encrypted hostnames. Among the 806 encrypted instances, the model only recognized 578 (71.7%) of them. Furthermore, the authors found that more than 60% of the 183 apps that have legitimate sharing connections also contain illegal transmissions inside for ad or analytics purposes. Interestingly, they also found a weather application that only transmits usersâ location data to ad servers.	n/a
[160]	Study	MLW	***Focus.*** Authors propose NSDroid, a time-efficient malware multi-classification approach based on neighborhood signature in local function call graphs. ***Method.*** NSDroid constructs the function call graph of the Android application and leverages the function call local structure, which is represented by the neighborhood signature, to compare the similarity with the trained malicious application families. Evaluation of several ML models has been performed. ***Results.*** NSDroid is based on SVM with an AUC score of 0.993. SVM has an excellent classification performance in TPR, precision, recall, and F-values in all experiments. Although the performance of RF on RocArea is the highest, SVM achieves the highest on PRCArea.	E

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[169]	Study	MLW	***Focus.*** A malware detection system that transforms malware files into images and classifies the image representation with CNN. ***Results.*** The dataset used for the experiment is provided by Korea University from the Andro-dumpsys study. Such a dataset consists of 906 malicious binary files from 13 malware families, including smishing and spy applications. ***Results.*** The results show that the resnet50 network has better performance when the input image is in RGB color space, and the MLP classifier performs better with grayscale images.	E
[164]	Study	PP	***Focus.*** This paper suggests the use of automatic topic modeling for large-scale corpora of privacy policies using unsupervised learning techniques. ***Method.*** Authors make use of the OPP-115 and the ACL/-COLING 2014 Datasets for training a Latent Dirichlet Allocation model. Then they asked to an expert lawyer to validate the extracted topics against GDPR. ***Results.*** The proposed method, based on Latent Dirichlet Allocation topic modeling, provides a means for monitoring the evolution of privacy policies over time and for reflecting changes in their content, in terms of the topics being addressed, given a specific set of regulations or following a regulatory change.	N
[124]	Frmwrk	GenLk	***Focus.*** Proposal of a text mining based method to infer the purpose of sensitive data access by Android apps. ***Method.*** The key idea authors propose here, is to extract multiple features from app code and then use those features to train a ML classifier for purpose inference. They present the design, implementation, and evaluation of two complementary approaches to infer the purpose of permission use, first using purely static analysis, and then using primarily dynamic analysis. ***Results.*** The proposed classifier is able to infer the purpose of 138 code instances out of 153. Authors also find that over 60% of call stacks in the evaluation were due to third-party libraries, most of which are ad libraries. In addition, more than 70% of app analyzed have obfuscated code which does not impact on the framework performance.	N
[136]	Tool	GenLk	***Focus.*** Authors present Android Privacy Assistant, a practical privacy protection system that reveals privacy disclosure and provides fine-grained privacy information modifications to balance privacy and data usability. ***Method.*** Authors divide a string of request into its component words as feature sets and use such features to feed ML methods. ***Results.*** The proposed solution uses C5.0 Decision Tree classifier to identify privacy leakages.	N

Continued on next page

Table 3 – *continued from previous page*

Ref.	**Type**	**Threat**	**Highlights**	**Prot.**
[161]	Frmwrk	MLW	***Focus.*** The article presents MLDroid, a web-based framework which helps to detect malware from Android devices. ***Method.*** Authors download a total of more than 50,000 malware and goodware .apk files from public repositories. Then they extract relevant features by gathering API calls and permission for each .apk and applying well-known feature ranking methods (such as PCA, gain-ratio selecion, info-gain). Lastly, they validate several ML methods to find the best one and/or the best combination among them by comparing the achieved F1-score. ***Results.*** The empirical result reveals that model developed by considering all the four distinct machine learning algorithms parallelly, i.e., deep learning algorithm, farthest first clustering, Y-MLP and nonlinear ensemble DT approach) and rough set analysis as a feature subset selection algorithm, achieved the highest detection rate of 98.8% to detect malware from real-world apps.	E
[162]	Tool	MLW	***Focus.*** An autonomous host-based intrusion detection system for Android mobile devices ***Method.*** The proposed approach needs to analyse different kinds of features at the device level: the total CPU usage, memory consumption, outgoing/incoming network traffic, battery level/volt- age/temperature, number or running processes/services, the status of screen (on/off), total number of opened TCP sockets, total installed applications, and total number of outgoing calls and SMSs. Then it employs several ML methods to detect anomalies due to malware. ***Results.*** The results show that the proposed intrusion detection system achieves a very promising accuracy of above 0.9983, reaching up to 1. All classifiers tested achieve 100% accuracy except NaiveBayes.	E
[152]	Tool	GenLk	***Focus.*** Mining smartphone users privacy perception on app markets ***Method.*** The authors exploit user reviews on the Google Play Store as a relevant source in order to extract and quantify privacy relevant claims associated with apps. ***Results.*** Based on ML, Natural Language Processing and sentiment analysis techniques, the developed tool named MARS detects privacy relevant reviews and categorizes them into a pre-identified list of privacy threats in the context of mobile apps. LR shows the best score with F1-score of 0.9279 in finding all the threats of an app on the Google Play Store. Interestingly, health-based apps are sometimes underestimated by the users. As compared with other popular app categories such as Lifestyle, users are not well-aware of the potential negative consequences of using privacy invasive health-based apps.	N

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[173]	Frmwrk	PP	***Focus.*** Scalable system to help analyze and predict Android appsâ compliance with privacy requirements. ***Method.*** In a preliminary analysis on a set of 17,991 Android apps they check whether they have a privacy policy. Then, for the 9,295 apps that have one they apply machine learning classifiers to analyze policy content based on a human-annotated corpus of 115 policies. Leveraging static analysis the authors investigate the actual data practices occurring in the appsâ code. ***Results.*** A novel combination of machine learning and static analysis techniques to analyze apps' potential non-compliance with privacy requirements. With a failure rate of 0.4%, a mean F-1 score of 0.96, and a mean analysis time of 6.2 seconds per app the approach makes large-scale app analyses for legally relevant data practices feasible and reliable. Mapping the policy to the app analysis results they identify and analyze privacy requirement inconsistencies between policies and apps. The system enables app publishers to identify potentially privacy-invasive practices in their apps before they are published.	N
[167]	Study	MLW	***Focus.*** Malware detection on Android. ***Method.*** In this paper, the authors evaluate an evolutionary system named MOCDroid [472] and a co-evolutionary system named ArmsRace [473], against a state-of-the-art rule-based system, namely Assemblyline, an open-source tool to detect and analyze malware. The evolutionary systems are trained on well-established malicious Android datasets, as well as benign applications, before being tasked with identifying large, unknown datasets. ***Results.*** MOCDroid is very effective on older malicious datasets such as Genome and Drebin, and benign datasets sourced from F-Droid and Google Play. In training, these detectors routinely score above 90% accuracy. ArmsRace is similarly effective on the aforementioned malicious and benign sets, but also manages a respectable 89% accuracy when evaluating the newer UNB Malicious samples. These results show that ArmsRace may be well suited for detecting new variants of malware. Furthermore, the results show that both the ArmsRace and MOCDroid systems are competitive with Assemblyline.	N
[148]	Frmwrk	MLW	***Focus.*** Android malware detection system using deep CNN. ***Method.*** The malware detection method uses a CNN to process the raw Dalvik bytecode of an Android application. Malware classification is performed based on static analysis of the raw opcode sequence from a disassembled program. Features indicative of malware are automatically learned by the network from the raw opcode sequence thus removing the need for hand-engineered malware features. The CNN is trained end-to-end to jointly learn appropriate features and to perform classification, thus removing the need to explicitly enumerate millions of n-grams during training. ***Results.*** This work shows that the application of CNN to the field of malware analysis provides good performance in comparison with other state-of-art techniques, and has been validated in four different Android malware datasets. The system is capable of simultaneously learning to perform feature extraction and malware classification given only the raw opcode sequences of a large number of labeled malware samples. The system can be implemented to run on the GPU of mobile devices.	E

Continued on next page

Table 3 – ***continued from previous page***

Ref.	**Type**	**Threat**	**Highlights**	**Prot.**
[143]	Tool	MLW	***Focus.*** Proposal of *ServiceMonitor*, a lightweight host-based detection system that dynamically detects malicious applications directly on mobile devices. ***Method.*** ServiceMonitor reconstructs the fine-grained behavior of applications based on their interaction with system services (i.e.SMS manager, camera, wifi networking, etc). ServiceMonitor monitors the way applications request system services in order to build a statistical Markov chain model to represent what and how system services are used. Afterwards, ServiceMonitor use this Markov chain as a feature vector to classify the application behavior into either malicious or benign using the RF method. ***Results.*** The authors evaluated ServiceMonitor using a dataset of 8034 malware and 10024 benign applications and obtaining 96.7% of accuracy rate and negligible overhead and performance penalty.	E
[127]	Tool	MLW	***Focus.*** In this paper, the authors propose DroidLight, a lightweight Intrusion Detection System which can detect zero-day malware efficiently and effectively. ***Method.*** The authors designed an algorithm for DroidLight that is based on one class classification and probability distribution analysis. For each smartphone application, the classification model learns its normal CPU usage and network traffic pattern. The model flags an intrusion alert if there is any significant deviation from the normal pattern. ***Results.*** By deploying three self-developed malwares, authors performed realistic evaluation of DroidLight, i.e. the evaluation was performed on a real device while a real user was interacting with it. Evaluation results demonstrate that DroidLight can detect smartphone malwares with accuracy ranging from 93.3% to 100% while imposing only 1.5% total overhead on device resources.	N
[156]	Tool	MLW	***Focus.*** The authors propose *DroidSieve*, an Android malware classifier based on static analysis that is fast, accurate, and resilient to obfuscation. ***Method.*** DroidSieve exploits obfuscation-invariant features and artifacts introduced by obfuscation mechanisms used in malware. At the same time, these purely static features are designed for processing at scale and can be extracted quickly. ***Results.*** For malware detection, authors achieve up to 99.82% accuracy with zero false positives; for family identification of obfuscated malware, they achieve 99.26% accuracy at a fraction of the computational cost of state-of-the-art techniques.	N

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[147]	Frmwrk	GenLk	***Focus.*** In this paper, the authors present an enhanced context-aware resource usage control system named *EasyPrivacy* to provide adaptive resource usage control. ***Method.*** In order to achieve a high accuracy in potentially malicious apps identification, authors utilize both conventional static features and n-gram opcodes (n-opcodes) of apps to detect suspicious apps installed on smartphones using ML methods and the ensemble of them. In addition, to facilitate users that have little domain knowledge in Android to configure policies reasonably, the authors have developed an application named *TipsTool*, which can guide users to configure policies step by step. ***Results.*** In order to get a reliable result, the authors test this system in a real environment that porting our customized Android OS and installing TipsTool to a Galaxy S4 smartphone. In the first experiment, they applied their modifications to the Android Nougat to implement our resource usage control system. Then they evaluated the functionality of our system by porting the modified Android OS to a Galaxy S4 smartphone. Results show that users can easily protect their privacy on smartphones through this system.	N
[131]	Frmwrk	GenLk	***Focus.*** In this paper, the authors propose a permission induced risk model, iOS Application analyzer and Behavior Classifier (iABC), for iOS devices to detect privacy violations arising due to granting permissions during installation of applications. ***Method.*** The system includes a two-layer process comprising of static and dynamic analysis. It uses reverse engineering to extract permission variables from applications and computes a risk score for each application using ranking algorithms. The approach considers application's category as a key feature for detecting malicious applications while computing static risk score. Different ML methods were employed to evaluate 1,150 applications. ***Results.*** The empirical results show that the proposed model gives detection rate of 97.04%.	N
[135]	Tool	MLW	***Focus.*** The authors present an Android malware detection and family identification approach, *RevealDroid*, that operates without the need to perform complex program analyses or to extract large sets of features. ***Method.*** The selected features leverage categorized Android API usage, reflection-based features, and features from native binaries of apps. ***Results.*** The authors assess RevealDroid for accuracy, efficiency, and obfuscation resilience using a large dataset consisting of more than 54,000 malicious and benign apps. The experiments show that RevealDroid achieves an accuracy of 98% in detection of malware and an accuracy of 95% in determination of their families.	E
[145]	Study	MLW	***Focus.*** The authors present two ML aided approaches for static analysis of Android malware. ***Method.*** The first approach is based on permissions and the other is based on source code analysis utilizing a bag-of-words representation model. The permission-based model is computationally inexpensive, and is implemented as the feature of OWASP Seraphimdroid Android app that can be obtained from Google Play Store. ***Results.*** The evaluations of both approaches indicate an F-score of 95.1% and 89% for the source code-based classification and permission-based classification models, respectively.	E

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[168]	Tool	MLW	***Focus.*** The authors present Malware Recomposition Variation (MRV), an approach that conducts semantic analysis of existing malware to systematically construct new malware variants for malware detectors to test and strengthen their detection signatures/models. ***Method.*** The authors use two variation strategies (i.e., malware evolution attack and malware confusion attack) following structures of existing malware to enhance feasibility of the attacks. Upon the given malware, they conduct semantic-feature mutation analysis and phylogenetic analysis to synthesize mutation strategies. Based on these strategies, the authors perform program transplantation to automatically mutate malware bytecode to generate new malware variants. ***Results.*** The authors evaluate MRV approach on actual malware variants, and also performed an empirical evaluation on 1,935 Android benign apps and 1,917 malware. Results show that MRV produces malware variants that can have high likelihood to evade detection while still retaining their malicious behaviors. They also propose and evaluate three defense mechanisms to counter MRV.	E
[140]	Study	CyAtt	***Focus.*** Aiding Android application developers in assessing the security and privacy risk associated with Android applications by using static code metrics as predictors. ***Method.*** Authors investigate how effectively static code metrics that are extracted from the source code of Android applications, can be used to predict security and privacy risk of Android applications. The authors collected 21 static code metrics of 1,407 Android applications, and use the collected static code metrics to predict security and privacy risk of the applications. ***Results.*** SVM method shows precision of 0.83, therefore it can be used effectively to predict security and privacy risk of Android applications.	–
[144]	Study	CyAtt	***Focus.*** On-device transformation of sensor data to be shared. ***Method.*** They formulate the anonymization problem using an information-theoretic approach and propose a new multi-objective loss function for training deep autoencoders. This loss function helps minimizing user-identity information as well as data distortion to preserve the application-specific utility. The training process regulates the encoder to disregard user-identifiable patterns and tunes the decoder to shape the output independently of users in the training set. ***Results.*** The trained autoencoder can be deployed on a mobile or wearable device to anonymize sensor data even for users who are not included in the training dataset. Data from 24 users transformed by the proposed anonymizing autoencoder lead to a promising trade-off between utility and privacy, with an accuracy for activity recognition above 92% and an accuracy for user identification below 7%.	N

Continued on next page

Table 3 – ***continued from previous page***

Ref.	Type	Threat	Highlights	Prot.
[157]	Study	GenLk	***Focus.*** Detecting personally identifiable information leaks. ***Method.*** The authors take a network centric approach. Since, by definition, personal information is transmitted from mobile apps over the network interface towards remote servers, network traffic is a natural vantage point for detection and control of such leaks. They develop *AntShield*, an approach that intercepts outgoing network packets on the device, and employs a hybrid string matching and classification approach to detect leaks of personally identifiable information. ***Results.*** The authors evaluate the performance of several ML methods using datasets that were collected and analyzed from scratch. They also report preliminary results that show that collaboration among users can further improve classification accuracy, thus motivating crowdsourcing and/or distributed learning of privacy leaks.	N
[221]	Tool	PP	***Focus.*** A privacy policy summarization tool, named *PrivacyGuide*. ***Method.*** The tool is inspired by the European Union GDPR and based on ML and a word embedding technique, *StringToWordVector* (to represent the text of PPs) provided by WEKA software. ***Results.*** Results show that PrivacyGuide is able to classify PP content into eleven privacy aspects with a weighted average accuracy of 74% and further shed light on the associated risk level with an accuracy of 90%.	N
[129]	Frmwrk	MLW	***Focus.*** Android malware detection system with a novel feature selection method. ***Method.*** The authors consider different importances of the features associated with their contributions to the classification problem as well as their manipulation costs, and present a novel feature selection method (named SecCLS) to make the classifier harder to be evaded. To improve the system security while not compromising the detection accuracy, the authors further propose an ensemble learning approach (named SecENS) by aggregating the individual classifiers that are constructed using the proposed feature selection method SecCLS. ***Results.*** Authors develop a system called SecureDroid which integrates the proposed methods (i.e., SecCLS and SecENS) to enhance security of ML based Android malware detection. Comprehensive experiments on the real sample collections from Comodo Cloud Security Center are conducted to validate the effectiveness of SecureDroid against adversarial Android malware attacks by comparisons with other alternative defense methods. The proposed secure-learning paradigm can also be readily applied to other malware detection tasks.	E

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[153]	Frmwrk	MLW	***Focus.*** The authors introduce Significant Permission IDentification (SigPID), a malware detection system to scale the detection for a large bundle of apps. ***Method.*** The system is based on permission usage analysis. SigPID utilizes ML methods to classify different families of malware and benign apps. ***Results.*** The evaluation finds that only 22 permissions are significant. Authors then compare the performance of such an approach, using only 22 permissions, against a baseline approach that analyzes all permissions. The results indicate that when a SVM is used as the classifier, they can achieve over 90% of precision, recall, accuracy, and F-measure, which are about the same as those produced by the baseline approach while incurring the analysis times that are 4â32 times less than those of using all permissions. Compared against other state-of-the-art approaches, SigPID is more effective by detecting 93.62% of malware in the dataset and 91.4% unknown/new malware samples.	E
[159]	Study	Spam	***Focus.*** Development of an anti-spam app system. ***Method.*** Through a systematic crawl of a popular app market and by identifying apps that were removed over a period of time, the authors propose a method to detect spam apps solely using app metadata available at the time of publication. They first propose a methodology to manually label a sample of removed apps, according to a set of checkpoint heuristics that reveal the reasons behind removal. The authors then map the identified heuristics to several quantifiable features and show how distinguishing these features are for spam apps. They adopt AdaBoost for early identification of spam apps using only the metadata of the apps. ***Results.*** The proposed ML method achieves an accuracy of over 95% with precision varying between 85% and 95% and recall varying between 38% and 98%. The authors further show that a limited number of features, in the range of 10â30, generated from app metadata is sufficient to achieve a satisfactory level of performance. On a set of 180,627 apps that were present at the app market during our crawl, the classifier predicts 2.7% of the apps as potential spam. Finally, the authors perform additional manual verification and show that human reviewers agree with 82% of the classifier predictions.	E
[146]	Frmwrk	MLW	***Focus.*** Malware detection system at a scale. ***Method.*** The authors introduce a streaminglized ML-based framework, StormDroid: *(i)* The core of StormDroid is based on ML, enhanced with a novel combination of contributed features that the authors observed over a fairly large collection of data set; and *(ii)* the authors streaminglize the whole detection process to support large-scale analysis, yielding an efficient and scalable technique that observes app behaviors statically and dynamically. ***Results.*** StormDroid has been evaluated on roughly 8,000 applications, and the combination of contributed features improves detection accuracy by almost 10% compared with state-of-the-art antivirus systems; when run in parallel, the streaminglized process further improves efficiency rate by approximately three times than a single thread.	E

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[166]	Study	GenLk	***Focus.*** Longitudinal study on contextuality behind privacy decisions of users to build an automatic system capable to predict their decisions. ***Method.*** The authors performed a longitudinal 131-person field study collecting real-world Android usage data in order to explore whether we could infer usersâ future privacy decisions based on their past privacy decisions, contextual circumstances surrounding applicationsâ data requests, and usersâ behavioral traits. Also, their phones periodically prompted them to make privacy decisions when applications used sensitive permissions, and they logged their decisions. Overall, participants wanted to block 60% of these requests. Then, the authors designed a ML based approach to automatically predict how users would respond to prompts based on SVM with rbf kernel. ***Results.*** The authors show that the approach can accurately predict users' privacy decisions 96.8% of the time.	NE
[139]	Frmwrk	GenLk	***Focus.*** Privacy-preserving framework for activity recognition. ***Method.*** This framework relies on a ML method to efficiently recognise the user activity pattern while limiting the risk of re-identification of users from biometric patterns that characterizes each individual. To achieve that, the authors first deeply analysed different features extraction schemes in both temporal and frequency domain. They show that features in temporal domain are useful to discriminate user activity while features in frequency domain lead to distinguish the user identity. On the basis of this observation, they second design a novel protection mechanism that processes the raw signal on the user's smartphone and transfers to the application server only the relevant features unlinked to the identity of users. ***Results.*** The authors extensively evaluate the framework with a reference dataset: results show an accurate activity recognition (87%) while limiting the re-identifation rate (33%). This represents a slightly decrease of utility (9%) against a large privacy improvement (53%) compared to state-of-the-art baselines.	–
[130]	Study	GenLk	***Focus.*** Identifying critical discrepancies between developer-described app behavior and permission usage. ***Method.*** Authors combine state-of-the-art techniques in natural language processing and ML. They design a CNN for text classification that captures the relevance of words and phrases in app descriptions in relation to the usage of dangerous permissions. The proposed system predicts the likelihood that an app requires certain permissions and can warn about descriptions in which the requested access to sensitive user data and system features is textually not represented. ***Results.*** The authors evaluate the solution on 77,000 real-world app descriptions and find that they can identify individual groups of dangerous permissions with a precision between 71% and 93%.	N

Continued on next page

Table 3 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[142]	Frmwrk	MLW	***Focus.*** Android malware detection system that is designed to be resilient to feature-unaware perturbations without retraining. ***Method.*** First, authors consider only a subset of the codebase of a given app, both for precision and performance aspects. Their implementation focuses exclusively on the loops contained in a given app. The authors hypothesize, and empirically verify, that the code contained in apps' loops is enough to precisely detect malware. This provides the additional benefits of being less prone to noise and errors, and being more performant. The second idea is to build a feature space by extracting a set of labels for each loop, and by then considering each unique combination of these labels as a different feature: the combinatorial nature of this feature space makes it prohibitively difficult for an attacker to influence the feature vector and avoid detection, without access to the specific model used for classification. ***Results.*** The authors assembled these techniques into a prototype, called LoopMC, which can locate loops in applications, extract features, and perform classification, without requiring source code. They used LoopMC to classify about 20,000 benign and malicious applications. While focusing on a smaller portion of the program may seem counterintuitive, the results of these experiments are surprising: the system achieves a classification accuracy of 99.3% and 99.1% for the Malware Genome Project and VirusShare datasets respectively, which outperforms previous approaches.	E

Internet of Things In Table 4 we show the identified studies in the area of IoT.

Table 4: Identified studies in the area of Internet of Things. GenLk = General data leakage, Dis = Disclosure, ImgLk = Image leakage, AnUsr = anomalous users, LocLk = location leakage, CyAtt = Cyber attack, HLk = health leakage, MLW = malware, SpLk = speech leakage, E = Enforcement, N = Nudge.

Ref.	Type	Threat	Highlights	Prot.
[120]	Study	CyAtt	***Focus.*** Intrusion detection system based on LSTM ***Method.*** Evaluation of traditional ML models versus LSTM on the NSL- Knowledge Discovery Data mining (NSL-KDD) dataset ***Results.*** LSTM outperform other models. The overall accuracy on validation data was 99.51%, the F-Score was 99.43% and the accuracy on test data was 86.99%.	E

Continued on next page

Table 4 – ***continued from previous page***

Ref.	**Type**	**Threat**	**Highlights**	**Prot.**
[121]	Study	MLW	***Focus.*** Using ML approaches for webshell detection in Internet of things environments ***Method.*** (1) A dataset including 1551 malicious PHP webshells and 2593 normal PHP scripts are collected for IoT server security experiments. (2) Authors study term frequency inverse document frequency (TF-IDF), opcode and combined Opcode-TF-IDF feature extraction methods for data preprocessing. They train six types of ML models, which are K-Means, MLP, NB, DT, SVM, and KNN. Then, these models are combined through ensembles. (3) Feature clustering analysis based on PCA is performed to analyze the dataset. ***Results.*** RF is more suitable for lightweight scenario due its efficiency. Although it requires more substantial computing resources and longer computing time, the Voting method achieves the maximum Recall score of 99.57% and maximum F-score of 98.32%. The most relevant features are *echo*, *send* and *return*.	E
[112]	Frmwrk	MLW	***Focus.*** Authors propose EveDroid, a scalable and event-aware Android malware detection system, which exploits the behavioral patterns in different events to effectively detect new malware based on the insight that events can reflect appsâ possible running activities. ***Method.*** Authors propose to use event group to describe appsâ behaviors in event level, which can capture higher level of semantics than in API level. In event group, they adopt function clusters to represent behaviors in each event so that behaviors hidden in events can still be captured as time goes on, which enables EveDroid to detect new malware in the event level. The function clusters can generalize API calls into vectors based on their API composition to capture new API calls. ***Results.*** MLP achieves up to 99.8% F-score on a dataset of 14956 benign and 28848 malicious Android apps released.	E
[122]	Study	CyAtt	***Focus.*** Proposing an innovative concept drift adaptive method to improve the accuracy of anomaly detection, which fully considers time influence to change the sample distribution along timeline ***Method.*** Authors aim at the phenomenon of concept drift in time-series data and propose an effective sampling method which is different from other existing conventional methods. The proposed method benefits from a large number of time-series data generated in IoT environment of smart home, which can improve various indicators to verify the weakening influence of concept drift by the proposed anomaly detection method. ***Results.*** Authors present a LSTM neural network that adds time factor and employs a novel smooth activation function, which can enhance the performance of multi-classification for anomaly detection achieving accuracy precision and recall close to 100%, above to other methods like a traditional neural network and SVM.	n/a

Continued on next page

Table 4 – ***continued from previous page***

Ref.	**Type**	**Threat**	**Highlights**	**Prot.**
[123]	Study	MLW	***Focus.*** Malware detection in mobile environments based on Autoencoders ***Results.*** This work represents the sequences of API calls invoked by malware during their execution as sparse matrices looking like images (API-images), which can be used as fingerprints of the malwareâs behavior over time. The dataset used during the experimental evaluation includes samples downloaded by the Malgenome and Contagio Minidump datasets. More recent malware have been also downloaded by the VirusShare site. The benign apps have been collected from Playdrone dataset. Other samples have been also collected from Google Play store. They compare traditional ML models performance versus a novel autoencoder. ***Results.*** Autoencoders based on API-image works better than all the others model experimented with F-score 0.97 and accuracy 0.94	E
[108]	Study	GenLk	***Focus.*** Computationally model and predict usersâ privacy preferences in IoT. ***Method.*** Authors survey 172 participants in a simulated campuswide IoT environment about their privacy preferences regarding hypothetical personal information tracking scenarios. Then, they cluster the scenarios based on the survey responses, arriving at four clusters with distinct associated privacy preferences. Based on the clustering results, authors uncover contextual factors that induce privacy violations in IoT. Finally, they build ML models to predict usersâ privacy decisions, using both contextual information and the corresponding cluster membership as training data. ***Results.*** DT predicts privacy preferences with 0.77 accuracy	N
[109]	Frmwrk	CyAtt	***Focus.*** Applying ML and Parallel Data Processing for cyber attack detection in IoT ***Method.*** Authors experiment different joint application of methods of ML and parallel data processing. Models compared include SVM, KNN, NB, MLP and DT. ***Results.*** First, the most accuracy was demonstrated by SVM, KNN, and DT. The usage of various methods of combining basic classifiers increases even more the accuracy of detection of attacks, and the greatest degree of accuracy increases with soft voting. Secondly, the best training time was shown by KNN, NB, and DT. However, the KNN and NB classifiers are much inferior to the others in testing time. The best testing time is associated to MLP.	E

Continued on next page

Table 4 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[117]	Study	GenLk	***Focus.*** Proposal of functional requirements for privacy-aware systems to facilitate well-informed privacy decision-making in IoT, which results in conservative and confident decisions that enjoy high consistency ***Method.*** Researchers first conducted an online survey on Amazon Mechanical Turk (N = 488) to collect peopleâs privacy decision-making about diverse IoT service scenarios that they may encounter in their everyday lives. Then, they created 180 realistic scenarios while varying underlying contextual factors of the IoT service, such as location, purpose, relationships between collectible sensor data and inferable personal information, and data privacy policies (e.g., how data will be protected, retained, and shared). Lastly, they used several classifiers to assess the possibility of predicting and suggesting privacy decisions. ***Results.*** ML experiments also revealed that individuals overall privacy awareness is the most important feature when predicting their privacy decisions. Researchers verified that RF model trained on privacy decisions made with confidence can produce highly accurate privacy recommendations for users (AUC of 87%).	N
[118]	Tool	CyAtt	***Focus.*** Proposal of an intelligent architecture that integrates Complex Event Processing technology and ML in order to detect different types of IoT cyber attacks in real time. ***Method.*** The proposed architecture relies on a middleware that creates the data connection between IoT networks and both the Complex Event Processing engine and ML methods, as well as to allow the Complex Event Processing engine to communicate with data consumers such as trusted computers. In this way, the Complex Event Processing engine receives both network packet events as a result of preprocessing IoT sensing data, and prediction network packet events generated upon training ML methods. By analyzing and correlating these events through the use of event patterns, the engine is capable of detecting IoT security attacks in real time. ***Results.*** The proposed intelligent architecture combining Complex Event Processing and ML is capable of easily managing dynamic patterns for detecting IoT cyber attacks. Therefore, using linear regression and SVR to calculate expected pattern values enables the detection of new attacks in real time.	E
[115]	Tool	HLk	***Focus.*** A recommendation approach for user privacy preferences in the fitness domain ***Method.*** Researchers first present a fitness data privacy model that they defined to represent usersâ privacy preferences in a way that is unambiguous, compliant with the GDPR, and able to represent both the user and the third party preferences. They collect a dataset using different scenarios in the fitness domain in order to identify privacy profiles by applying ML techniques. Then they examine different personal tracking data and user traits which can potentially drive the recommendation of privacy profiles to the users. ***Results.*** A set of privacy-setting recommendation strategies with different guidance styles are designed based on the resulting profiles. Using K-means, one can capture the preferences of various users with a higher level of accuracy.	N

Continued on next page

Table 4 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[110]	Study	CyAtt	***Focus.*** Attack detecion based on distributed learning. ***Method.*** The authors design and implement a deep learning based distributed attack detection mechanism, which reflects the underlying distribution features of IoT. ***Results.*** The experiment has shown the successful adoption of ML to cybersecurity, and designed and implemented the system for attack detection in distributed architecture of IoT applications such as smart cities. The evaluation process has employed accuracy, the detection rate, false alarm rate, etc. as performance metrics to show the effectiveness of deep models over shallow models. The experiment has demonstrated that distributed attack detection can better detect cyber-attacks than centralized algorithms because of the sharing of parameters which can avoid local minima in training.	E
[114]	Study	CyAtt	***Focus.*** A technique to automatically detect consumer IoT attack traffic. ***Method.*** The authors use IoT-specific network behaviors (e.g., limited number of endpoints and regular time intervals between packets) to inform feature selection. ***Results.*** They demonstrate a high accuracy DDoS detection in IoT network traffic with a variety of ML methods. Such methods successfully identify attack traffic with an accuracy higher than 0.999. The authors found that RF, KNN, and MLP were particularly effective.	E

Web In Table 5 we show the identified studies in the area of Web.

Table 5: Identified studies in the Web domain. GenLk = General data leakage, Dis = Disclosure, ImgLk = Image leakage, AnUsr = anomalous users, LocLk = location leakage, HLk = health leakage, CyAtt = Cyber attack, Phish = phishing, MLW = malware, PP = Privacy Policy, E = Enforcement, N = Nudge.

Ref.	Type	Threat	Highlights	Prot.
[196]	Study	Phish	***Focus.*** Determining if a target website is a phishing website or not, based on the standard deinition of a phishing website from literature. ***Method.*** The he intuition that the domain name of phishing websites is the tell-tale sign of phishing and holds the key to successful phishing detection. Authors design features that model the relationships, visual as well as statistical, of the domain name to the key elements of a phishing website, which are used to snare the end-users. Then use such features as input for ML models. ***Results.*** XGBoost achieves accuracy score above 0.98. The first set of experiments were conducted on a prepared dataset and the second set of experiments were conducted on live unknown phishing dataset from OpenPhish.com	N

Continued on next page

Table 5 – *continued from previous page*

Ref.	**Type**	**Threat**	**Highlights**	**Prot.**
[225]	Study	PP	***Focus.*** Using automated and semi-automated methods for extracting from privacy policies the data practice details that are salient to Internet usersâ interests ***Method.*** Authors crowdsourced answers to questions about privacy policies. They use the OPP-115 Corpus of privacy policies to train and test classifiers in order to label policy text with common themes. The ML models used are LR, SVM, CNN. ***Results.*** It appears that data practices can be reliably extracted from privacy policies through crowdsourcing, and it is thus a viable mechanism to provide the data required for privacy policy analysis. Automated policy annotation is feasibile both on sentences and fragments with best results obtained with SVM and fragments.	N
[214]	Study	GenLk	***Focus.*** Studying minified and obfuscated JavaScript code in the Web. ***Method.*** The datase analyzed is composed of 967,149 scripts (424,023 unique) from the top 100,000 websites. The regular code examples are a subset of a corpus of 150,000 JavaScript files available in literature, which consists of human-written, non-transformed code from open-source project. Authors then consider seven minification tools (e.g. Google Closure Compiler, YUI compressor). As training data for a specific classifier, they randomly sampled 10,000 files from the corpus of regular files and apply transformations tools to these files. Lastly, at the core of the study there is a neural network-based classifier trained to identify whether obfuscation or minification have been applied and if yes, using what tools. ***Results.*** Neural network-based classifier can identify highly accurate (95%-100%) JavaScript code with particular properties. Authors find that code transformations are very widespread, affecting 38% of all scripts. Studying which code gets obfuscated, authors find that obfuscation is common in certain website categories, e.g., adult content, and that some obfuscated scripts trigger suspicious behavior, such as likely fingerprinting and timing attacks. Furthermore multiple obfuscated scripts access privacy sensitive APIs and use dynamic code loading for apparently unknown purposes. They also find that the most popular obfuscation technique is to load code at runtime via eval.	E
[195]	Study	Phish	***Focus.*** Phishing Detection through semantic analysis ***Method.*** The authors extract a series of semantic features through word embedding techniques (word2vec) to describe the features of phishing sites, and further fuse them with other multi-scale statistical features to construct a more robust phishing detection model. In this study, AdaBoost, Bagging, RF, and SMO were chosen to implement the learning and testing of phishing detection models. ***Results.*** The experimental results on the dataset show that the majority of phishing websites are effectively identified by only mining the semantic features of word embedding. AdaBoost obtains the best results in terms of F-score on a single ML model experiment (0.993.). Fusion models have significant advantages in F-score, false positive rate and error rate, indeed the F-score of fusion model reaches 0.998 but the improvement is minimal.	E

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[193]	Frmwrk	Phish	***Focus.*** Development of a framework, named "Fresh-Phish", for creating ML data for phishing websites detection. ***Results.*** Authors exploited 30 different website static features. They build a large labeled dataset and analyze several ML classifiers against this dataset to determine which is the most accurate. They analyze not just the accuracy of the technique, but also how long it takes to train the model. ***Results.*** Results of MLP are close to 90% of AUC. On the employed dataset the results show that the extra training required of MLP over SVM was not worth the minuscule gains in accuracy.	E
[228]	Study	Phish	***Focus.*** Developing techniques that can detect phishing websites automatically and handle zero-day phishing attacks ***Method.*** Authors make use of a CNN module to extract character-level spatial feature representations of URLs; meanwhile, they employ an attention-based hierarchical Recurrent Neural Network (RNN) module to extract word-level temporal feature representations of URLs. They then fuse these feature representations via a three-layer CNN to build accurate feature representations of URLs, on which authors train a phishing URL classifier. ***Results.*** The proposed model PhishingNet(CNN+RNN) outperforms all the baselines across all the evaluation metrics, demonstrating the effectiveness of the proposed approach.	E
[217]	Study	Phish	***Focus.*** Delopment of an explainable/understandable phishing detection model ***Method.*** Authors explore deep learning approaches and build several Recurrent Neural Network (RNN) models that only use lexical features of URLs for detecting phishing attacks. ***Results.*** All the RNN models achieve a similar phishing detection accuracy. Combining the predictions of the RNN models into a simple ensemble model can further help eliminate most false positives and false negatives; the RNN models can well capture the relevant features including protocol, IP address, special characters, and different parts in URLs that were manually extracted and used in the traditional ML approach for phishing detection.	E
[218]	Tool	PP	***Focus.*** Describing a method for the automated detection of opt-out choices in privacy policy text and their presentation to users through a web browser extension. ***Method.*** Authors create two corpora of opt-out choices, which enable the training of classifiers to identify opt-outs in privacy policies using word based features such as embeddings and TF-IDF. ***Results.*** The approach based on LR achieves a precision of 0.93 and a recall of 0.9 in detecting opt-out choices. Authors then introduce Opt-Out Easy, a web browser extension designed to present available opt-out choices to users as they browse the web which has been appreciated by the participants of a user study in terms of usability.	N

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[219]	Tool	GenLk	***Focus.*** Hybrid and lightweight detection of third party tracking: Design, implementation, and evaluationpropose the use of a hybrid mechanism exploiting blacklisting and ML for the automatic identification of privacy intrusive services requested while browsing Web pages ***Method.*** Authors download a dataset of Web resources from Alexa's top 16000. They use handcrafted features for modeling JavaScript and other HTTP requests. They compared the performance of several classifiers in literature including RF, MLP, KNN, SVM in the task of classifying resources into tracking and not tracking. ***Results.*** Google Chrome plugin working with MLP can effectively protect users from tracking on the Web. The plugin shows great efficiency and effectiveness when compared with privacy tools available on the market.	E
[209]	Frmwrk	CyAtt	***Focus.*** Proposal of a privacy-preserving technique to enable the data creator to compress data via collaborative learning maximizing utility and preserving privacy. ***Method.*** The idea is to define *Compressive privacy* (CP) technique, which enable the data creator to encrypt data using compressive-and-lossy transformation and hence protects the userâs personal privacy while delivering the intended (classification) capability. The objective of CP is to learn what kind of compressed data may enable classification/recognition of, say, face or speech data while concealing the original face images or speech content from malicious attackers. ***Results.*** Authors use a discriminant component analysis (DCA) method for: *(i) Utility-driven DCA*: the rank of the signal subspace is limited by the number of classes, and so DCA can effectively support classification using a relatively small dimensionality; *(ii) Desensitized PCA*: by incorporating a signal-subspace ridge into DCA, it leads to a variant especially effective for extracting privacy-preserving components; *(iii) Desensitized K-means*: since the revelation of the K-means cluster structure could leak sensitive information, it is safer to perform clustering on a desensitized PCA subspace.	E
[224]	Study	MLW	***Focus.*** Design, development and evaluation of COUGAR, a system to reduce high-dimensional malware behavioural data, and optimize clustering behaviour using a multi-objective genetic algorithm. ***Method.*** Authors employed the Endgame Malware BEnchmark for Research (EMBER) dataset from Endgame [474], designed as a benchmark dataset for researchers and including features from over 2,000,000 Windows Portable Executables. Then, the authors utilize evolutionary multiobjective optimization, which is able to produce a sequence of parameters for the following clustering algorithms: DBSCAN, OPTICS, and K-means. ***Results.*** The results indicate that each clustering algorithm is capable of producing at least satisfactory results, and highlight many highquality clusters. Specifically, OPTICS identifies the greatest number of clusters, while DBSCAN achieves the lowest summed sum of squared errors. Finally, in our real-world scenario, the analyst successfully identified the WannaCry samples [475] with the help of COUGARâs predictions, which got 12 of 13 labels correct.	E

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[197]	Study	GenLk	***Focus.*** DNS backscatter detection. ***Method.*** The idea is to use ML (DT, RF, and SVM) to classify the originators of each malicious activity into broad groups (spammers, scanners, and several types of commercial activity). The authors study long-term accuracy of the algorithm and show that malicious and benign originators exhibit different activity behaviors and evolution. They also study how different training strategies affect classification accuracy and show that it is necessary to aggressively adapt training according to changing world. ***Results.*** They use backscatter and classification to examine world-wide scanning activity from three sources and up to nine months. They also examine nine months of data, showing that there is a continuous background of scanning, and identifying increased scanning following announcements of vulnerabilities.	–
[223]	Study	CyAtt	***Focus.*** Detection of abnormal HTTP queries ***Method.*** This work introduces a dynamic feature extraction using the well-known Autoencoders, which can operate directly on the raw HTTP request and perform the classification using different parts of both the request body and header. They also propose to use ensemble learning methods that combine the outcomes of the individual Autoencoders with the aim to improve the overall system performance. ***Results.*** The authors demonstrate that XGBoost is able to achieve a much higher detection rate with respect to other state-of-the-art solutions. Specifically, such approach is demonstrated to achieve a detection rate of up to 99.84%, overcoming other supervised and unsupervised learning methods, such as Regularized Deep Autoencoders, DT and LSTM.	N
[211]	Study	CyAtt	***Focus.*** Malicious domain names detection system capable to cope with attackers that can virtually spawn an infinite number of shadowed domains. ***Method.*** They identify a set of novel features that uniquely characterize domain shadowing by analyzing the deviation from their apex domains and the correlation among different apex domains. Building upon these features, they train a classifier and apply it to detect shadowed domains on the daily feeds of VirusTotal, a large open security scanning service. ***Results.*** The result is a novel system, called *Woodpecker* which applied on the daily feeds of VirusTotal, obtain 287,780 reports, of which 127,561 are confirmed as shadowed domains with a set of heuristics (most of the remaining ones are about malicious apex domains). The measurement of the characteristics of these shadowed domains indicates that they exhibit quite different properties from conventional malicious domains, and thus existing systems can hardly detect the shadowed domains.	E

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[200]	Study	MLW	***Focus.*** Automatic malicious domain names detection which overcome blacklisting mechanisms. ***Method.*** The authors exploit a dataset of technical mail list, i.e., the input are mail messages. They develop components to parse the raw data, extract a set of salient features, and build a model to classify whether the domains are involved in malicious activities. ***Results.*** The authors evaluate their system on real-world mailing lists. The ground truth information is collected from both public blacklists and manual checking. The authors show that the proposed system is able to identify malicious domains accurately, resulting in a 94% detection rate with zero false positive rate. The results show that Gossip provides earlier detection compared to existing blacklists.	E
[199]	Study	MLW	***Focus.*** Evaluating the applicability of adversarial examples to the case of malware detection. ***Method.*** Starting from a binary indicator vector X to represent an application, the authors apply a MLP to solve the malware classification task. They use a rectifier as the activation function for each hidden neuron in the network. As output, they employ a softmax layer for normalization of the output probabilities. To train the network, the standard gradient descent and standard dropout were used. ***Results.*** The authors evaluate the training of the MLP based malware detector and adversarial example-induced misclassification of inputs on it. With the MLP they achieve classification performance matching stateof-the-art from the literature. Using the augmented adversarial crafting algorithm they then manage to mislead this classifier for 63% of all malware samples.	E
[220]	Study	PP	***Focus.*** Summarize PPs to better inform users. ***Method.*** The authors propose a ML based approach to summarize the rather long PPs into short and condensed notes following a risk-based approach and using the European Union GDPR aspects as assessment criteria. Once the main categories were defined, a set of PP were analyzed by a group of privacy experts, as a result, three risk levels for each of the categories were determined. ***Results.*** Experiments were carried out on four model (NB, SVM, DT, and RF). While NB and SVM seem to be robust in the available training and testing data, DT and RF showed low precision. In summary, the benefits of the the proposed approach are two fold: *(i)* supporting users in assessing or interpreting PPs; *(ii)* providing actionable insights for service providers with respect to their PPs.	N
[205]	Study	MLW	***Focus.*** Malware classification and visualization. ***Method.*** The authors used a database of 9,339 samples of malwares from 25 families. They calculated the GIST descriptor for grayscale malware images as input for KNN method. ***Results.*** The KNN method was trained and evaluated many times to reach a score of 97%, which is very close to other results found in literature.	E

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[192]	Study	MLW	***Focus.*** Assessing main issues of ML methods for network security in real-world applications. ***Method.*** The authors highlight two primary issues: inaccurate ground truth and a highly non-stationary data distribution. To demonstrate and understand the effect that these pitfalls have on popular ML methods, the authors design and carry out experiments that show how six common methods perform when confronted with real network data. ***Results.*** With the experimental results, the authors identify the situations in which certain classes of methods underperform on the task of encrypted malware traffic classification. They offer concrete recommendations for practitioners given the real-world constraints outlined. From an algorithmic perspective, the authors find that the RF ensemble method outperformed competing methods. More importantly, feature engineering was decisive; they found that iterating on the initial feature set, and including features suggested by domain experts, had a much greater impact on the performance of the classification system. Their analysis is based on millions of TLS encrypted sessions collected over 12 months from a commercial malware sandbox and two geographically distinct, large enterprise networks.	E
[227]	Study	MLW	***Focus.*** Malware detection based on images. ***Method.*** Authors extract important byte sequences in malware samples by application of CNN to images converted from binary data. They also combine a technique called the attention mechanism into CNN, which shows regions having higher importance for classification in the image. The extracted region with higher importance can provide useful information for human analysts who investigate the functionalities of unknown malware samples. ***Results.*** Results of the evaluation experiment using malware dataset show that the proposed method provides higher classification accuracy than a conventional method. Furthermore, analysis of malware samples based on the calculated attention map confirmed that the extracted sequences provide useful information for manual analysis.	–
[210]	Study	MLW	***Focus.*** Effectively applying ML with no domain knowledge for malware detection. ***Method.*** In this work, the authors develop a new SHWeL feature vector representation of malware byte sequences, by extending the recently proposed Lempel-Ziv Jaccard Distance. These SHWeL vectors improve upon LZJD's accuracy, outperform byte n-grams, and allow to build efficient algorithms for both training (a weakness of byte n-grams) and inference (a weakness of LZJD). Furthermore, the new SHWeL method also allows the researchers to directly tackle the class imbalance problem, which is common for malware-related tasks. ***Results.*** Compared to existing methods like SMOTE, SHWeL provides significantly improved accuracy while reducing algorithmic complexity to $O(N)$. Because the approach is developed without the use of domain knowledge, it can be easily re-applied to any new domain where there is a need to classify byte sequences.	–

Continued on next page

Table 5 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[222]	Study	Phish	***Focus.*** Phishing detection system. ***Method.*** The authors perform a measurement study on squatting phishing domains where the websites impersonate trusted entities not only at the page content level but also at the web domain level. To search for squatting phishing pages, they scanned five types of squatting domains over 224 million DNS records and identified 657K domains that are likely impersonating 702 popular brands. Then they build a ML method to detect phishing pages under the squatting domains. A key novelty is that the ML method is built on a careful measurement of evasive behaviors of phishing pages in practice. The authors introduce new features from visual analysis and optical character recognition to overcome the heavy content obfuscation from attackers. ***Results.*** In total, authors discovered and verified 1,175 squatting phishing pages. The authors show that these phishing pages are used for various targeted scams, and are highly effective to evade detection. More than 90% of them successfully evaded popular blacklists for at least a month.	N
[207]	Frmwrk	GenLk	***Focus.*** Using ML methods to reduce information communicated over the Internet. ***Method.*** The idea is making the sent data noisy without compromise the accuracy of the system that should use the information. An undisciplined addition of noise can significantly reduce the accuracy of inference, rendering the service unusable. Authors propose *Shredder*, an end-to-end framework, that, without altering the topology or the weights of a pre-trained network, learns additive noise distributions that significantly reduce the information content of communicated data while maintaining the inference accuracy. ***Results.*** Experimentation with six real-world NN from text processing and image classification shows that Shredder reduces the mutual information between the input and the communicated data to the cloud by 74.70% compared to the original execution while only sacrificing 1.58% loss in accuracy. On average, Shredder also offers a speedup of 1.79Ã over Wi-Fi and 2.17Ã over LTE compared to cloud-only execution when using an off-the-shelf mobile GPU (Tegra X2) on the edge.	N
[212]	Study	MLW	***Focus.*** A Hardware-Assisted Malware Detection (HMD) technique. ***Method.*** The idea is to detect patterns of malicious applications based on microarchitectural features captured by modern microprocessorsâ Hardware Performance Counters (HPCs). The authors propose *StealthMiner*, a specialized time series machine learning approach to accurately detect embedded malware at run-time using branch instructions feature, the most prominent microarchitectural feature. ***Results.*** The results indicate that StealthMiner can detect embedded malware at runtime with 94% detection performance on average with only one HPC feature, outperforming the detection performance of state-of-the-art HMD methods by 42%.	E

General In Table 6 we show the identified studies in a not specified area, which we named General.

Table 6: Identified studies in a not specified area, which are more general-domain oriented. GenLk = General data leakage, Dis = Disclosure, ImgLk = Image leakage, AnUsr = anomalous users, LocLk = location leakage, HLk = health leakage, CyAtt = Cyber attack, E = Enforcement, N = Nudge.

Ref.	Type	Threat	Highlights	Prot.
[87]	Study	LocLk	***Focus.*** A Data Mining Approach to Assess Privacy Risk in Human Mobility Data ***Method.*** The idea is to train classifiers to capture the relation between individual mobility patterns and the level of privacy risk of individuals. Starting from a dataset of around 1 million GPS tracks, researchers extract individual mobility patterns and compute the privacy risk level associated with vehicles according to the repertoire of reidentification attacks. They then train ML models and use them to determine (in polynomial time) the privacy risk level of previously unseen vehicles whose data were not used in the learning phase, based just on their individual mobility patterns. ***Results.*** RF is accurate in classifying the privacy risk level of unseen individuals in the two urban areas. The predictions are particularly accurate in classifying the lowest and the highest levels of privacy risk, allowing an immediate distinction between safe individuals and risky individuals. The second remarkable result is that the classifiers built on one urban area are effective when used to determine the privacy risk level of individuals in the other urban area. This suggests that the predictive models are able to infer rather general relationships between mobility patterns and privacy risk, which are independent of the number of individuals, the width of the geographic area, and the length of the period of observation	n/a
[101]	Framework	Spam	***Focus.*** Proposal of an anti-spam framework that fully relies on unsupervised methodologies through a multi-algorithm clustering approach. ***Method.*** The proposed framework examines only the domain and header related information found in email headers. A novel method of feature reduction using an ensemble of âunsupervisedâ feature selection algorithms has also been investigated in this study. ***Results.*** Out of six different clustering algorithms used, Spectral and K-means demonstrated acceptable performance while OPTICS projected the optimum clustering with an average of 3.5% better efficiency than Spectral and K-means, validated through a range of validations processes. The average balanced accuracy for the optimum three algorithms has been found to be about 94.91%	E
[102]	Frmwrk	SpLk	***Focus.*** Proposal of a protocol to decrease the risks of correlation of speech content and speakerâs voice (child, teenager, and adult) in speech data publishing. ***Method.*** Authors formalize the correlation between speech content and speakerâs voice and regard it as a new kind of privacy leakage risk. Then, they utilize the ML and optimize speech data sanitization considering the defined risks of privacy disclosure and data utility loss. Finally, simulation results validate the effectiveness of the proposed protocol. ***Results.*** Results indicate that the method based on DT can lead to a lower attack success rate when compared with others available in literature.	E

Continued on next page

Table 6 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[89]	Study	HLk	***Focus.*** Proposal of an event-level privacy protection, and develop a feature ablation method to protect event-level privacy in electronic medical records. ***Method.*** Researchers selected 13 sensitive diseases. They construct various sensitive disease patient cohorts and the non-sensitive disease patient cohort (consisting of patients without any diagnosis in the sensitive disease category) by querying Northwestern Medicine Enterprise Data Warehouse. Then then use the patientsâ laboratory test results, medications and procedures as their features for ML models. ***Results.*** LR has been used for the feature ablation test. Authors find that the sensitive diagnoses can be divided into three categories: (1) five diseases have fast declining identifiability (AUC below 0.6 with less than 400 features excluded); (2) seven diseases with progressively declining identifiability (AUC below 0.7 with between 200 and 700 features excluded); and (3) one disease with slowly declining identifiability (AUC above 0.7 with 1,000 features excluded).	E
[103]	Frmwrk	GenLk	***Focus.*** A framework that allows user to independently use their private data to locally train a one-class reconstructive adversarial network that represents their training data. ***Method.*** Authors used a Reconstructive Adversarial Networks that enable users to perform private training locally without a public pre-trained feature extractor. Then, for the prediction phase, a multi-class classifier is aggregated in the cloud by leveraging a third party (a regulator) that aids in the computation while learning nothing about user data ***Results.*** The framework effectiveness was proved.	E
[105]	Study	CyAtt	***Focus.*** Detecting the DNS tunnel conveniently and effectively. ***Method.*** Authors present a novel method that uses Autoencoder to learn latent representation of different datasets. ***Results.*** Their experiments show how autoencoders can automatically learn the concept of semantic similarity among features of normal traffic. Indeed, the obtained results show that the recall rate can exceed 0.9834 on the labeled dataset and 0.9313 on the SINGH-data.	E
[96]	Study	CyAtt	***Focus.*** Proposal of a new dimension reduction-based method for privacy preservation. ***Method.*** The idea is to generate dimension-reduced data for performing ML tasks in the cloud so to prevent a strong adversary from reconstructing the original data. Authors employed three different face image datasets (AT&T, YaleB, and CelebA) for the experiments. ***Results.*** The experiments with proposed model, based on Generative Adversarial Network, show that when the number of dimensions is reduced to seven, it can achieve the accuracies of 79%, 80%, and 73% respectively and the reconstructed images are not recognizable to naked human eyes.	E

Continued on next page

Table 6 – *continued from previous page*

Ref.	Type	Threat	Highlights	Prot.
[107]	Study	GenLk	***Focus.*** Studying data leaks over encrypted data with ML methods. ***Method.*** The authors built a supervised learning, classification, based model that aims to find the related topic (category) of each ciphertext, which is encoded using several encryption algorithms. The pre-processing part of this research only consists of extracting the features using character n-grams and representing them using TF-IDF. Additionally, they encoded the data using the same key for each encryption and then fed the training data to the ML methods (NB, RF, LR, and SVM). Based on the range of n-grams and the TF-IDF scores, the goal was to explore whether the ML methods would be able to find a pattern on the ciphertext that could generalize and use it for the unseen data to predict the related topic (class) of that ciphertext. ***Results.*** Evaluations show that data leakage decreases as the encryption algorithms get stronger, even though the data leakage is never zero percent. The performance of the model on the modern encryption algorithms is above the baseline for a given dataset. This means that some data leakage exist in all of the implementations of these algorithms. These results are gathered on two datasets where one is binary and the other is multiclass. The same trends are observed on both the binary and the multiclass datasets. This seems to indicate that the proposed approach generalizes well.	–
[97]	Tool	ImgLk	***Focus.*** Automatic privacy recommendation for images ***Method.*** A new tool called *iPrivacy* is developed for releasing the burden from users on setting the privacy preferences when they share their images for special moments. Specifically, a deep multi-task learning algorithm is developed to jointly learn more representative CNNs and more discriminative DT methods, so that authors can achieve fast and accurate detection of large numbers of privacy-sensitive object classes. ***Results.*** To evaluate the performance of iPrivacy system, the authors have conducted extensive experimental studies on real-world images. All the experiments are conducted on a parallel set that comprises about 800,000 social images and their privacy settings. The experimental results have demonstrated both efficiency and effectiveness of the proposed approach.	N
[100]	Study	ImgLk	***Focus.*** Configuring successful privacy settings for social image sharing. ***Method.*** For achieving more compact representation of image content sensitiveness (privacy), two approaches are developed: 1) a MLP is adapted to extract 1024-D discriminative deep features; and 2) a deep multiple instance learning algorithm is adopted to identify 280 privacy sensitive object classes and events. Futhermore, users on the social network are clustered into a set of representative social groups to generate a discriminative dictionary for user trustworthiness characterization. Finally, both the image content sensitiveness and the user trustworthiness are integrated to train a tree classifier to recommend fine-grained privacy settings for social image sharing. ***Results.*** Experimental studies have demonstrated both efficiency and effectiveness of the proposed algorithms.	N

.2 Word and sentence embedding

In the Natural Language Processing (NLP) field there have been introduced several techniques to understand the meaning of words or sentences for purposes ranging from question answering [476] to sentiment analysis [477]. In the last years, word embedding has established itself as one of the most popular representation methods of document vocabulary [478, 479]. Among its capabilities we can cite that of capturing the context of a word in a document, semantic and syntactic similarity, relation with other words, and so on. Word2vec [480, 481] is the most popular technique in this field. They use the conditional probability $P(w|c)$ to predict the target word w based on its context c. They have been used for a variety of tasks, e.g., finance-relating text mining [482] and question answering [483].

The success of neural network methods for computing word embeddings, has motivated the proposal of several methods for generating semantic embeddings of longer pieces of text, such as sentences and paragraphs. They are methods to embed a full sentence into a n-dimensional vector space. These sentence embeddings retain some friendly properties, as they inherit features from their underlying word embeddings [484].

Google has developed its own sentence embedding method, named *Universal Sentence Encoder*, which is capable of dealing with a large number of tasks in NLP [485]. First, they developed it in English language, and second, they expanded such a method for accounting more than ten languages, including Italian, German, Spanish, etc [456], and they made it available for developers and researchers through Tensorflow Hub[2]. These multilingual version specifically embeds text from 16 languages into a single semantic space using a multi-task trained dual-encoder that learns tied representations using, in turn, translation based bridge tasks [486]. The Universal Sentence Encoder has proved to show good performance with minimal amounts of supervised training data [485]; it takes in input a variable-length text, and the output

[2] https://bit.ly/36BSS52

is a 512-dimensional vector. In this work, we used the multilingual Universal Sentence Encoder to represent clauses in Terms of Service online (see Chapter 7) and, for text messages and tweets a user is about to disseminate on the Internet (see Chapter 5).

.3 ML for awareness: a closer look to search keywords

In the following, we show the search keywords used for downloading from Twitter the datasets used in Chapter 5. Specifically, we show the keywords used for downloading tweets about the topics "racism", "religion", and "sexual orientation".

Racism keywords

Keywords here are combined with an `AND` operator with the set of common *hate-words* in English, available via *EnglishClub*[a]. Then we have applied an `OR` operator between each pair.

Keywords: africa, african, african american, african-american, african-americans, africans, afro, american, asian, asians, black, black people, blacks, brown, caucasian, china, chinese, chink, coon, cracker, crackers, dago, desi, hick, hillbilly, hindu, honky, india, indian, indians, italian, japanese, kike, korean, limey, mexican, native, native american, negro, negroes, negroid, negros, nig, nigga, niggas, nigger, niggers, niglet, redneck, redneck, white boy, white people, white power, white supremacy, white trash, wigger, wog, wop, yid

[a] `https://www.englishclub.com/ref/Slang/Insulting/`

Religion keywords

Keywords here are combined with an `OR` operator.

Keywords: Allah, Jah, Jehovah, Jesus, Messiah, Yahweh, agnostic, almighty, atheism, atheist, atheists, athiest, baptist, baptize, belief, bible, buddha, buddhism, catholic, catholicism, catholics, christ, christian, christianity, christians, church, churches, conservative, deism, deist, deistic, deity, devotion, devotional, dharma, divine, divinity, god, god-fearing, goddess, godhead, godly, gods, hebrew, hell, hindu, hinduism, holiness, holy, islam, jainism, jesus, jew, jewish, jews, judaism, krishna, lord, mandir, messiah, moksha, mormon, muslim, myth, mythology, numen, omnipotent, pietism, piety, pious, pope, poped, pray, prayer, prayerful, priest, redeemer, religion, religiosity, religious, reverend, ritual, sacred, saint, saint, satan, shiva, sikhism, sin, spiritual, spirituality, superstition, temple, theism, theist, torah, totem, tutelary, vatican, vishnu, worship, yiddish

Sexual Orientation keywords

Keywords here are combined with an `OR` operator.

Keywords: agender, androgyne, androgynous, androgyny, asexual, bi, bi-sexual, bicurious, bisexous, bisexual, bisexual chic, bisexuality, butch, cis, cisgendered, cishet, cissexual, dike, drag queen, dyke, effeminate, enbyfriend, fag, faggot, faggots, fags, female, female-to-male, feminine, feminism, femme, ftm, gay, gender, gender binary, gender identity, genderfucked, genderqueer, giri, girlfag, guydyke, hermaphrodite, hertosexual, hetero, heterosexual, heterosexuality, hetrosexual, homo, homos, homosexual, homosexuality, ladyboy, lesbian, lesbians, lesbo, lgbt, logisexual, male, man, manly, masculine, men, metrosexual, metrosexuals, montisexual, mtf, neologist, neutrois, nitro sexual, no gender, nongender, not gay, omni, omnisexual, on the pull, pan, pansexual, pansexuality, pomosexual, pretty boy, queen, queer, queers, relationship, retrosexual, sex change, sexism, sexless, sexual orientation, sexuality, shemale, straight, tranny, trans, transexual, transfag, transgender, transgendered, transgenderqueer, transguy, transman, transphobia, transsexual, transvestite, trisexual, ubersexual, woman, women, zoosexual

.4 ML for awareness: further performance metrics about Topic module

In this section, we detail the performance obtained by each ML method compared in Chapter 5 for the Topic module, that is Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), AdaBoost (Ada), and MultiLayer Perceptron (MLP).

Metric	**Topic**							
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
Accuracy	0.962							
Precision	0.959	0.985	0.975	0.961	0.969	0.929	0.955	0.959
Recall	0.95	0.995	1.00	0.995	0.935	0.920	0.96	0.94
F1-score	0.955	0.990	0.988	0.978	0.951	0.925	0.958	0.949

Table 7: Results for all classification performance metrics obtained by **RF** in the Topic module. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = Racism, Rel = Religion, SO = Sexual Orientation.

Metric	**Topic**							
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
Accuracy	0.99							
Precision	0.989	1.000	0.98	1.000	0.990	0.975	0.989	0.995
Recall	0.985	1.000	1.000	0.995	1.000	0.980	0.975	0.985
Fscore	0.987	1.000	0.99	0.997	0.995	0.977	0.982	0.989

Table 8: Results for all classification performance metrics obtained by **SVM** in the Topic module. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = Racism, Rel = Religion, SO = Sexual Orientation.

Metric	Topic							
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
Accuracy	0.983							
Precision	0.979	0.995	0.985	0.995	0.990	0.951	0.984	0.985
Recall	0.975	1.000	1.00	0.995	0.990	0.975	0.945	0.985
Fscore	0.977	0.997	0.992	0.995	0.990	0.963	0.964	0.985

Table 9: Results for all classification performance metrics obtained by **KNN** in the Topic module. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = Racism, Rel = Religion, SO = Sexual Orientation.

Metric	Topic							
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
Accuracy	0.961							
Precision	0.974	0.995	0.966	0.995	0.956	0.921	0.931	0.954
Recall	0.950	0.980	0.995	0.965	0.97	0.935	0.95	0.945
Fscore	0.962	0.987	0.980	0.979	0.963	0.928	0.941	0.949

Table 10: Results for all classification performance metrics obtained by **Ada** in the Topic module. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = Racism, Rel = Religion, SO = Sexual Orientation.

Metric	Topic							
	P	**H**	**J**	**T**	**G**	**Rac**	**Rel**	**SO**
Accuracy	0.982							
Precision	0.969	0.985	0.978	0.961	0.979	0.929	0.965	0.959
Recall	0.96	0.995	1.00	0.995	0.945	0.920	0.970	0.94
Fscore	0.965	0.990	0.989	0.978	0.961	0.925	0.978	0.959

Table 11: Results for all classification performance metrics obtained by **MLP** in the Topic module. P = Politics, H = Health, J = Job, T = Travel, G = General, Rac = Racism, Rel = Religion, SO = Sexual Orientation.

Bibliography

[1] B. H. Bratton, *The stack: On software and sovereignty.* MIT press, 2016.

[2] J. Danaher, M. J. Hogan, C. Noone, R. Kennedy, A. Behan, A. De Paor, H. Felzmann, M. Haklay, S.-M. Khoo, J. Morison *et al.*, "Algorithmic governance: Developing a research agenda through the power of collective intelligence," *Big Data & Society*, vol. 4, no. 2, p. 2053951717726554, 2017.

[3] E. Bayamlıoğlu and R. Leenes, "The ?rule of law?implications of data-driven decision-making: a techno-regulatory perspective," *Law, Innovation and Technology*, vol. 10, no. 2, pp. 295–313, 2018.

[4] J. R. Reidenberg, "Lex informatica: The formulation of information policy rules through technology," *Tex. L. Rev.*, vol. 76, p. 553, 1997.

[5] A. Cavoukian *et al.*, "Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada*, vol. 5, 2009.

[6] L. Lessig, *Code: And other laws of cyberspace.* ReadHowYouWant. com, 2009.

[7] R. Brownsword, "What the world needs now: Techno-regulation, human rights and human dignity," *Global governance and the quest for justice, Hart Publishing, Oxford*, pp. 203–34, 2004.

[8] S. Hassan and P. De Filippi, "The expansion of algorithmic governance: From code is law to law is code," *Field Actions Science*

Reports. The journal of field actions, no. Special Issue 17, pp. 88–90, 2017.

[9] "Cohubicol," https://www.cohubicol.com/.

[10] B. Van den Berg and R. E. Leenes, "Abort, retry, fail: scoping techno-regulation and other techno-effects," in *Human law and computer law: Comparative perspectives.* Springer, 2013, pp. 67–87.

[11] M. Hildebrandt, "Legal protection by design: objections and refutations," *Legisprudence*, vol. 5, no. 2, pp. 223–248, 2011.

[12] T. Kerikmäe and A. Rull, *The future of law and etechnologies.* Springer, 2016, vol. 3.

[13] M. Hildebrandt, "Algorithmic regulation and the rule of law," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 376, no. 2128, p. 20170355, 2018.

[14] P. De Filippi and S. Hassan, "Blockchain technology as a regulatory technology: From code is law to law is code," *arXiv preprint arXiv:1801.02507*, 2018.

[15] K. Yeung, "Algorithmic regulation: A critical interrogation," *Regulation & Governance*, vol. 12, no. 4, pp. 505–523, 2018.

[16] R. Leenes, "Framing techno-regulation: An exploration of state and non-state regulation by technology," *Legisprudence*, vol. 5, no. 2, pp. 143–169, 2011.

[17] S. D. Warren and L. D. Brandeis, "The right to privacy," *Harward Law Review*, vol. 4, no. 5, pp. 193–220, December 1890.

[18] A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, vol. 25, no. 1, p. 166, 1968.

[19] F. D. Schoeman, *Privacy and social freedom.* Cambridge university press, 1992.

[20] Y. Onn, M. Geva, Y. Druckman, A. Zyssman, R. L. Timor, I. Lev, A. Maroun, T. Maron, Y. Nachmani, Y. Simsolo *et al.*, "Privacy in the digital environment," *Haifa Center of Law & Technology*, pp. 1–12, 2005.

[21] Q. Wang, H. Xue, F. Li, D. Lee, and B. Luo, "# donttweet-this: Scoring private information in social networks," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 72–92, 2019.

[22] T. Zarsky, "The trouble with algorithmic decisions: An analytic road map to examine efficiency and fairness in automated and opaque decision making," *Science, Technology, & Human Values*, vol. 41, no. 1, pp. 118–132, 2016.

[23] J. Gomez, T. Pinnick, and A. Soltani, "School of information, uc berkeley," 2009.

[24] P. R. Clearinghouse, "Online privacy: Using the internet safely," 2012.

[25] D. Malandrino, V. Scarano, and R. Spinelli, "How increased awareness can impact attitudes and behaviors toward online privacy protection," in *2013 International Conference on Social Computing.* IEEE, 2013, pp. 57–62.

[26] L. Van Zoonen, "Privacy concerns in smart cities," *Government Information Quarterly*, vol. 33, no. 3, pp. 472–480, 2016.

[27] E. Zeng, S. Mare, and F. Roesner, "End user security and privacy concerns with smart homes," in *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*, 2017, pp. 65–80.

[28] J. Gu, Y. C. Xu, H. Xu, C. Zhang, and H. Ling, "Privacy concerns for mobile app download: An elaboration likelihood model perspective," *Decision Support Systems*, vol. 94, pp. 19–28, 2017.

[29] K. Degirmenci, "Mobile usersâ information privacy concerns and the role of app permission requests," *International Journal of Information Management*, vol. 50, pp. 261–272, 2020.

[30] J. Shaw, “The erosion of privacy in the internet era,” *Harvard Magazine*, pp. 38–43, 2009.

[31] D.-K. Kipker, “Privacy by default und privacy by design,” *Datenschutz und Datensicherheit-DuD*, vol. 39, no. 6, pp. 410–410, 2015.

[32] N. Lettieri, A. Guarino, D. Malandrino, and R. Zaccagnino, “Platform economy and techno-regulationÃ¢ÂÂexperimenting with reputation and nudge,” *Future Internet*, vol. 11, no. 7, p. 163, 2019.

[33] P. W. Singer and A. Friedman, *Cybersecurity: What everyone needs to know.* oup usa, 2014.

[34] R. V. G. Clarke and G. R. Newman, *Designing out crime from products and systems.* Criminal Justice Press Monsey, NY, 2005, vol. 18.

[35] R. Brownsword and K. Yeung, *Regulating technologies: Legal futures, regulatory frames and technological fixes.* Bloomsbury Publishing, 2008.

[36] C. Soghoian, “The history of the do not track header,” *Slight Paranoia, February*, 2012.

[37] R. H. Thaler and C. R. Sunstein, *Nudge: Improving decisions about health, wealth, and happiness.* Penguin, 2009.

[38] K. Yeung, “’hypernudge’: Big data as a mode of regulation by design,” *Information, Communication & Society*, vol. 20, no. 1, pp. 118–136, 2017.

[39] A. Gervais, A. Filios, V. Lenders, and S. Capkun, “Quantifying web adblocker privacy,” in *European Symposium on Research in Computer Security.* Springer, 2017, pp. 21–42.

[40] M. Kührer, C. Rossow, and T. Holz, “Paint it black: Evaluating the effectiveness of malware blacklists,” in *International Workshop on Recent Advances in Intrusion Detection.* Springer, 2014, pp. 1–21.

[41] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, "Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 379–396.

[42] J. A. Jupin, T. Sutikno, M. A. Ismail, M. S. Mohamad, S. Kasim, and D. Stiawan, "Review of the machine learning methods in the classification of phishing attack," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1545–1555, 2019.

[43] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[44] J.-h. Li, "Cyber security meets artificial intelligence: A survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1462–1474, 2018.

[45] D. Dasgupta, Z. Akhtar, and S. Sen, "Machine learning in cybersecurity: a comprehensive survey," *The Journal of Defense Modeling and Simulation*, p. 1548512920951275, 2020.

[46] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "Iot security techniques based on machine learning: How do iot devices use ai to enhance security?" *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 41–49, 2018.

[47] M. Moh and R. Raju, "Machine learning techniques for security of internet of things (iot) and fog computing systems," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2018, pp. 709–715.

[48] S. Imtiaz, R. Sadre, and V. Vlassov, "On the case of privacy in the iot ecosystem: A survey," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2019, pp. 1015–1024.

[49] M. Amiri-Zarandi, R. A. Dara, and E. Fraser, "A survey of machine learning-based solutions to protect privacy in the internet of things," *Computers & Security*, p. 101921, 2020.

[50] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in iot security: current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, 2020.

[51] I. Kayes and A. Iamnitchi, "Privacy and security in online social networks: A survey," *Online Social Networks and Media*, vol. 3, pp. 1–21, 2017.

[52] Q. Jamil and M. A. Shah, "Analysis of machine learning solutions to detect malware in android," in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*. IEEE, 2016, pp. 226–232.

[53] J. Zhang, C. Li, J. Ye, and G. Qu, "Privacy threats and protection in machine learning," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 531–536.

[54] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 1051–1052.

[55] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," *Information and Software Technology*, vol. 55, no. 7, pp. 1165–1199, 2013.

[56] A. Roehrs, C. A. Da Costa, R. da Rosa Righi, and K. S. F. De Oliveira, "Personal health records: a systematic literature review," *Journal of medical Internet research*, vol. 19, no. 1, p. e13, 2017.

[57] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed. O'Reilly Media, Inc., 2017.

[58] A. Chelli and M. Pätzold, "A machine learning approach for fall detection and daily living activity recognition," *IEEE Access*, vol. 7, pp. 38 670–38 687, 2019.

[59] X. Zhang, Y. Zhang, S. Wang, Y. Yao, B. Fang, and S. Y. Philip, "Improving stock market prediction via heterogeneous information fusion," *Knowledge-Based Systems*, vol. 143, pp. 236–247, 2018.

[60] D. Wong and S. Yip, "Machine learning classifies cancer," 2018.

[61] A. Cosimato, R. De Prisco, A. Guarino, D. Malandrino, N. Lettieri, G. Sorrentino, and R. Zaccagnino, "The conundrum of success in music: Playing it or talking about it?" *IEEE Access*, vol. 7, pp. 123 289–123 298, 2019.

[62] S. Lee, Y. Kim, H. Kahng, S.-K. Lee, S. Chung, T. Cheong, K. Shin, J. Park, and S. B. Kim, "Intelligent traffic control for autonomous vehicle systems based on machine learning," *Expert Systems with Applications*, vol. 144, p. 113074, 2020.

[63] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[64] Q. Wu, Q. Liu, Y. Zhang, and G. Wen, "Trackerdetector: A system to detect third-party trackers through machine learning," *Computer Networks*, vol. 91, pp. 164–173, 2015.

[65] L. Wang, *Support vector machines: theory and applications.* Springer Science & Business Media, 2005, vol. 177.

[66] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[67] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, "Finite state automata and simple recurrent networks," *Neural computation*, vol. 1, no. 3, pp. 372–381, 1989.

[68] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Artificial neural networks: concept learning*, 1990, pp. 112–127.

[69] A. E. Hoerl, R. W. Kannard, and K. F. Baldwin, "Ridge regression: some simulations," *Communications in Statistics-Theory and Methods*, vol. 4, no. 2, pp. 105–123, 1975.

[70] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[71] R. E. Schapire, "Explaining adaboost," in *Empirical inference*. Springer, 2013, pp. 37–52.

[72] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[73] K. Singh, H. K. Shakya, and B. Biswas, "Clustering of people in social network based on textual similarity," *Perspectives in Science*, vol. 8, pp. 570–573, 2016.

[74] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[75] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[76] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE, 2014, pp. 232–238.

[77] D. A. Reynolds, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, 2009.

[78] Y. Le Cun and F. Fogelman-Soulié, "Modèles connexionnistes de l'apprentissage," *Intellectica*, vol. 2, no. 1, pp. 114–143, 1987.

[79] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[80] F. F. Soulié, P. Gallinari, Y. Le Cun, and S. Thiria, "Automata networks and artificial intelligence," in *Centre National de Recherche Scientifique on Automata networks in computer science: theory and applications*, 1987, pp. 133–186.

[81] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic review in software engineering," *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, vol. 679, no. 05, p. 45, 2005.

[82] M. Petticrew and H. Roberts, *Systematic reviews in the social sciences: A practical guide.* John Wiley & Sons, 2008.

[83] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[84] A. Rajšp and I. Fister, "A systematic literature review of intelligent data analysis methods for smart sport training," *Applied Sciences*, vol. 10, no. 9, p. 3013, 2020.

[85] F. Scicchitano, A. Liguori, M. Guarascio, E. Ritacco, and G. Manco, "A deep learning approach for detecting security attacks on blockchain." in *ITASEC*, 2020, pp. 212–222.

[86] M. Bahrami, M. Singhal, and W.-P. Chen, "Learning data privacy and terms of service from different cloud service providers," in *2017 IEEE International Conference on Smart Cloud (SmartCloud).* IEEE, 2017, pp. 250–257.

[87] R. Pellungrini, L. Pappalardo, F. Pratesi, and A. Monreale, "A data mining approach to assess privacy risk in human mobility data," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 3, pp. 1–27, 2017.

[88] P. Arora and T. Chaspari, "Exploring siamese neural network architectures for preserving speaker identity in speech emotion classification," in *Proceedings of the 4th International Workshop on Multimodal Analyses Enabling Artificial Agents in Human-Machine Interaction*, 2018, pp. 15–18.

[89] C. Mao, Y. Zhao, M. Sun, and Y. Luo, "Are my ehrs private enough? event-level privacy protection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 16, no. 1, pp. 103–112, 2018.

[90] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, "Context-aware generative adversarial privacy," *Entropy*, vol. 19, no. 12, p. 656, 2017.

[91] L. Kopeykina and A. V. Savchenko, "Automatic privacy detection in scanned document images based on deep neural networks," in *2019 International Russian Automation Conference (RusAutoCon)*. IEEE, 2019, pp. 1–6.

[92] G. Kaur, Y. Malik, H. Samuel, and F. Jaafar, "Detecting blind cross-site scripting attacks using machine learning," in *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning*, 2018, pp. 22–25.

[93] S. Liu, J. Du, A. Shrivastava, and L. Zhong, "Privacy adversarial network: Representation learning for mobile data privacy," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–18, 2019.

[94] Y. Lin, X. Zhu, Z. Zheng, Z. Dou, and R. Zhou, "The individual identification method of wireless device based on dimensionality reduction and machine learning," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3010–3027, 2019.

[95] C. K. Ng, F. Jiang, L. Y. Zhang, and W. Zhou, "Static malware clustering using enhanced deep embedding method," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 19, p. e5234, 2019.

[96] H. Nguyen, D. Zhuang, P.-Y. Wu, and M. Chang, "Autogan-based dimension reduction for privacy preservation," *Neurocomputing*, vol. 384, pp. 94–103, 2020.

[97] J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan, "iprivacy: image privacy protection by identifying sensitive objects via deep multi-task learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1005–1016, 2016.

[98] H. Zuhair, A. Selamat, and O. Krejcar, "A multi-tier streaming analytics model of 0-day ransomware detection using machine learning," *Applied Sciences*, vol. 10, no. 9, p. 3210, 2020.

[99] M. Z. Alom, B. Carminati, and E. Ferrari, “Helping users managing context-based privacy preferences,” in *2019 IEEE International Conference on Services Computing (SCC)*. IEEE, 2019, pp. 100–107.

[100] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan, “Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing,” *IEEE transactions on information forensics and security*, vol. 13, no. 5, pp. 1317–1332, 2018.

[101] A. Karim, S. Azam, B. Shanmugam, and K. Kannoorpatti, “Efficient clustering of emails into spam and ham: The foundational study of a comprehensive unsupervised framework,” *IEEE Access*, vol. 8, pp. 154 759–154 788, 2020.

[102] G. Zhang, S. Ni, and P. Zhao, “Enhancing privacy preservation in speech data publishing,” *IEEE Internet of Things Journal*, 2020.

[103] A. S. Shamsabadi, A. Gascón, H. Haddadi, and A. Cavallaro, “Privedge: From local to distributed private training and prediction,” *IEEE Transactions on Information Forensics and Security*, 2020.

[104] M. Romanelli, K. Chatzikokolakis, and C. Palamidessi, “Optimal obfuscation mechanisms via machine learning,” in *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*. IEEE Computer Society, 2020, pp. 153–168.

[105] K. Wu, Y. Zhang, and T. Yin, “Tdae: Autoencoder-based automatic feature learning method for the detection of dns tunnel,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[106] M. Abdalla, M. Abdalla, F. Rudzicz, and G. Hirst, “Using word embeddings to improve the privacy of clinical notes,” *Journal of the American Medical Informatics Association*, vol. 27, no. 6, pp. 901–907, 2020.

[107] A. K. moghaddam and N. Zincir-Heywood, “Exploring data leakage in encrypted payload using supervised machine learning,” in

Proceedings of the 15th International Conference on Availability, Reliability and Security, 2020, pp. 1–10.

[108] H. Lee and A. Kobsa, "Privacy preference modeling and prediction in a simulated campuswide iot environment," in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom).* IEEE, 2017, pp. 276–285.

[109] A. Branitskiy, I. Kotenko, and I. B. Saenko, "Applying machine learning and parallel data processing for attack detection in iot," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[110] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[111] H. Mehrpouyan, I. M. Azpiazu, and M. S. Pera, "Measuring personality for automatic elicitation of privacy preferences," in *2017 IEEE Symposium on Privacy-Aware Computing (PAC).* IEEE, 2017, pp. 84–95.

[112] T. Lei, Z. Qin, Z. Wang, Q. Li, and D. Ye, "Evedroid: Event-aware android malware detection against model degrading for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6668–6680, 2019.

[113] M. Z. Alom, B. Carminati, and E. Ferrari, "Adapting users' privacy preferences in smart environments," in *2019 IEEE International Congress on Internet of Things (ICIOT).* IEEE, 2019, pp. 165–172.

[114] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW).* IEEE, 2018, pp. 29–35.

[115] O. R. Sanchez, I. Torre, Y. He, and B. P. Knijnenburg, "A recommendation approach for user privacy preferences in the fitness domain," *User Modeling and User-Adapted Interaction*, pp. 1–53, 2019.

[116] A. Cohen-Hadria, M. Cartwright, B. McFee, and J. P. Bello, "Voice anonymization in urban sound recordings," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[117] H. Lee and A. Kobsa, "Confident privacy decision-making in iot environments," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 27, no. 1, pp. 1–39, 2019.

[118] J. Roldán, J. Boubeta-Puig, J. L. Martínez, and G. Ortiz, "Integrating complex event processing and machine learning: An intelligent architecture for detecting iot security attacks," *Expert Systems with Applications*, vol. 149, p. 113251, 2020.

[119] N. T. Nguyen, T. T. Pham, T. X. Dang, M.-S. Dao, D.-T. Dang-Nguyen, C. Gurrin, and B. T. Nguyen, "Malware detection using system logs," in *Proceedings of the 2020 Intelligent on Intelligent Cross-Data Analysis and Retrieval Workshop*, 2020, pp. 9–14.

[120] S. M. Kasongo and Y. Sun, "A deep long short-term memory based classifier for wireless intrusion detection system," *ICT Express*, vol. 6, no. 2, pp. 98–103, 2020.

[121] B. Yong, W. Wei, K.-C. Li, J. Shen, Q. Zhou, M. Wozniak, D. Połap, and R. Damaševičius, "Ensemble machine learning approaches for webshell detection in internet of things environments," *Transactions on Emerging Telecommunications Technologies*, p. e4085, 2020.

[122] R. Xu, Y. Cheng, Z. Liu, Y. Xie, and Y. Yang, "Improved long short-term memory based anomaly detection with concept drift adaptive method for supporting iot services," *Future Generation Computer Systems*, 2020.

[123] G. DâAngelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and api-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, 2020.

[124] H. Wang, Y. Li, Y. Guo, Y. Agarwal, and J. I. Hong, "Understanding the purpose of permission use in mobile apps," *ACM*

Transactions on Information Systems (TOIS), vol. 35, no. 4, pp. 1–40, 2017.

[125] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Emulator vs real phone: Android malware detection using machine learning," in *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics*, 2017, pp. 65–72.

[126] H. Fu, Z. Zheng, S. Bose, M. Bishop, and P. Mohapatra, "Leaksemantic: Identifying abnormal sensitive network transmissions in mobile applications," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications.* IEEE, 2017, pp. 1–9.

[127] S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Droidlight: Lightweight anomaly-based intrusion detection system for smartphone devices," in *Proceedings of the 21st International Conference on Distributed Computing and Networking*, 2020, pp. 1–10.

[128] N. Bakhshinejad and A. Hamzeh, "A new compression based method for android malware detection using opcodes," in *2017 Artificial Intelligence and Signal Processing Conference (AISP).* IEEE, 2017, pp. 256–261.

[129] L. Chen, S. Hou, and Y. Ye, "Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 362–372.

[130] J. Feichtner and S. Gruber, "Understanding privacy awareness in android app descriptions using deep learning," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 203–214.

[131] A. J. Bhatt, C. Gupta, and S. Mittal, "iabc: Towards a hybrid framework for analyzing and classifying behaviour of ios applications using static and dynamic analysis," *Journal of Information Security and Applications*, vol. 41, pp. 144–158, 2018.

[132] N. V. Duc and P. T. Giang, "Nadm: Neural network for android detection malware," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, 2018, pp. 449–455.

[133] Y. Zhang, Y. Yang, and X. Wang, "A novel android malware detection approach based on convolutional neural network," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 2018, pp. 144–149.

[134] R. Kumar, Z. Xiaosong, R. U. Khan, J. Kumar, and I. Ahad, "Effective and explainable detection of android malware based on machine learning algorithms," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, 2018, pp. 35–40.

[135] J. Garcia, M. Hammad, and S. Malek, "Lightweight, obfuscation-resilient detection and family identification of android malware," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 26, no. 3, pp. 1–29, 2018.

[136] H.-Z. Tan, W. Zhao, and H.-H. Shen, "A context-perceptual privacy protection approach on android devices," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[137] P. Zegzhda, D. Zegzhda, E. Pavlenko, and G. Ignatev, "Applying deep learning techniques for android malware detection," in *Proceedings of the 11th International Conference on Security of Information and Networks*, 2018, pp. 1–8.

[138] M. Kakavand, M. Dabbagh, and A. Dehghantanha, "Application of machine learning algorithms for android malware detection," in *Proceedings of the 2018 International Conference on Computational Intelligence and Intelligent Systems*, 2018, pp. 32–36.

[139] T. Jourdan, A. Boutet, and C. Frindel, "Toward privacy in iot mobile devices for activity recognition," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 155–165.

[140] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile sensor data anonymization," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 49–58.

[141] Y. Lee, Y. Kim, S. Lee, J. Heo, and J. Hong, "Machine learning based android malware classification," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2019, pp. 300–302.

[142] A. Machiry, N. Redini, E. Gustafson, Y. Fratantonio, Y. R. Choe, C. Kruegel, and G. Vigna, "Using loops for malware classification resilient to feature-unaware perturbations," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 112–123.

[143] M. Salehi, M. Amini, and B. Crispo, "Detecting malicious applications using system services request behavior," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp. 200–209.

[144] A. Rahman, P. Pradhan, A. Partho, and L. Williams, "Predicting android application security and privacy risk with static code metrics," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 2017, pp. 149–153.

[145] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[146] S. Chen, M. Xue, Z. Tang, L. Xu, and H. Zhu, "Stormdroid: A streaminglized machine learning-based system for detecting android malware," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 377–388.

[147] B. Ren, C. Liu, B. Cheng, S. Hong, J. Guo, and J. Chen, "Easyprivacy: Context-aware resource usage control system for android platform," *IEEE Access*, vol. 6, pp. 44 506–44 518, 2018.

[148] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé *et al.*, "Deep android malware detection," in *Proceedings of the*

Seventh ACM on Conference on Data and Application Security and Privacy, 2017, pp. 301–308.

[149] X. Qin, F. Zeng, and Y. Zhang, "Msndroid: the android malware detector based on multi-class features and deep belief network," in *Proceedings of the ACM Turing Celebration Conference-China*, 2019, pp. 1–5.

[150] A. Mahindru and P. Singh, "Dynamic permissions based android malware detection using machine learning techniques," in *Proceedings of the 10th innovations in software engineering conference*, 2017, pp. 202–210.

[151] A. Seiderer, M. Dietz, I. Aslan, and E. André, "Enabling privacy with transfer learning for image classification dnns on mobile devices," in *Proceedings of the 4th EAI International Conference on Smart Objects and Technologies for Social Good*, 2018, pp. 25–30.

[152] M. Hatamian, J. Serna, and K. Rannenberg, "Revealing the unrevealed: Mining smartphone users privacy perception on app markets," *Computers & Security*, vol. 83, pp. 332–353, 2019.

[153] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[154] W. Li and Z. Liu, "Android malicious application detection method based on multi-class characteristics," in *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, 2019, pp. 157–161.

[155] M. M. H. Onik, C.-S. Kim, N.-Y. Lee, and J. Yang, "Personal information classification on aggregated android applicationâs permissions," *Applied Sciences*, vol. 9, no. 19, p. 3997, 2019.

[156] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "Droidsieve: Fast and accurate classification of obfuscated android malware," in *Proceedings of the Seventh ACM*

on Conference on Data and Application Security and Privacy, 2017, pp. 309–320.

[157] A. Shuba, E. Bakopoulou, and A. Markopoulou, "Privacy leak classification on mobile devices," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[158] F. Pierazzi, G. Mezzour, Q. Han, M. Colajanni, and V. Subrahmanian, "A data-driven characterization of modern android spyware," *ACM Transactions on Management Information Systems (TMIS)*, vol. 11, no. 1, pp. 1–38, 2020.

[159] S. Seneviratne, A. Seneviratne, M. A. Kaafar, A. Mahanti, and P. Mohapatra, "Spam mobile apps: Characteristics, detection, and in the wild analysis," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 1, pp. 1–29, 2017.

[160] P. Liu, W. Wang, X. Luo, H. Wang, and C. Liu, "Nsdroid: efficient multi-classification of android malware using neighborhood signature in local function call graphs," *International Journal of Information Security*, pp. 1–13, 2020.

[161] A. Mahindru and A. Sangal, "Mldroidâframework for android malware detection using machine learning techniques," *Neural Computing and Applications*, pp. 1–58, 2020.

[162] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, S. J. Shepherd, and R. A. Abd-Alhameed, "An autonomous host-based intrusion detection system for android mobile devices," *Mobile Networks and Applications*, vol. 25, no. 1, pp. 164–172, 2020.

[163] J. Zhang, X. Zhuang, and Y. Chen, "Android malware detection combined with static and dynamic analysis," in *Proceedings of the 2019 the 9th International Conference on Communication and Network Security*, 2019, pp. 6–10.

[164] D. Sarne, J. Schler, A. Singer, A. Sela, and I. Bar Siman Tov, "Unsupervised topic extraction from privacy policies," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 563–568.

[165] Z. Wang, G. Li, Y. Chi, J. Zhang, T. Yang, and Q. Liu, "Android malware detection based on convolutional neural networks," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019, pp. 1–6.

[166] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, "The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 1077–1093.

[167] Z. Wilkins and N. Zincir-Heywood, "Darwinian malware detectors: a comparison of evolutionary solutions to android malware," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 1651–1658.

[168] W. Yang, D. Kong, T. Xie, and C. A. Gunter, "Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 288–302.

[169] K. He and D.-S. Kim, "Malware detection with malware images using deep learning techniques," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 95–102.

[170] L. Suhuan and H. Xiaojun, "Android malware detection based on logistic regression and xgboost," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2019, pp. 528–532.

[171] X. Zhang and Z. Xu, "On the feasibility of automatic malware family signature generation," in *Proceedings of the First Workshop on Radical and Experiential Security*, 2018, pp. 69–72.

[172] V. M. Akman, R. A. Vural, and K. T. Koc, "Touchless authentication system using visual fingertip trajectories," *Electrica*, vol. 20, no. 2, pp. 143–153, 2020.

[173] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. M. Sadeh, S. M. Bellovin, and J. R. Reidenberg, "Automated analysis of privacy requirements for mobile apps." in *NDSS*, 2017.

[174] F. Raber, F. Kosmalla, and A. Krueger, "Fine-grained privacy setting prediction using a privacy attitude questionnaire and machine learning," in *IFIP Conference on Human-Computer Interaction*. Springer, 2017, pp. 445–449.

[175] P. Cappellari, S. A. Chun, and M. Perelman, "A tool for automatic assessment and awareness of privacy disclosure," in *Proceedings of the 18th Annual International Conference on Digital Government Research*, 2017, pp. 586–587.

[176] A. Almansour, G. Alsaeedi, H. Almazroui, and H. Almuflehi, "I-privacy photo: Face recognition and filtering," in *Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis*, 2020, pp. 131–141.

[177] A. Squicciarini, C. Caragea, and R. Balakavi, "Toward automated online photo privacy," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 1, pp. 1–29, 2017.

[178] P. Cappellari, S. A. Chun, and C. Costello, "Detecting and analyzing privacy leaks in tweets." in *DATA*, 2018, pp. 265–275.

[179] X. Chen, X. Song, R. Ren, L. Zhu, Z. Cheng, and L. Nie, "Fine-grained privacy detection with graph-regularized hierarchical attentive representation learning," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 4, pp. 1–26, 2020.

[180] N. Dhakal, F. Spezzano, and D. Xu, "Predicting friendship strength for privacy preserving: a case study on facebook," in *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 2017, pp. 1096–1103.

[181] Z. Shen, S. Fan, Y. Wong, T.-T. Ng, and M. Kankanhalli, "Human-imperceptible privacy protection against machines," in

Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1119–1128.

[182] W. B. Tesfay, J. Serna, and K. Rannenberg, "Privacybot: detecting privacy sensitive information in unstructured texts," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS).* IEEE, 2019, pp. 53–60.

[183] J. Zhang, B. Zhang, and J. Lin, "Recessive social networking: Preventing privacy leakage against reverse image search," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).* IEEE, 2019, pp. 211–219.

[184] Z. Xia, C. Liu, N. Z. Gong, Q. Li, Y. Cui, and D. Song, "Characterizing and detecting malicious accounts in privacy-centric mobile social networks: A case study," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2012–2022.

[185] T. Gao, J. Yang, W. Peng, L. Jiang, Y. Sun, and F. Li, "A content-based method for sybil detection in online social networks via deep learning," *IEEE Access*, vol. 8, pp. 38 753–38 766, 2020.

[186] A. Tonge and C. Caragea, "Dynamic deep multi-modal fusion for image privacy prediction," in *The World Wide Web Conference*, 2019, pp. 1829–1840.

[187] E. Aghasian, S. Garg, and J. Montgomery, "An automated model to score the privacy of unstructured informationâsocial media case," *Computers & Security*, vol. 92, p. 101778, 2020.

[188] L. Luceri, D. Andreoletti, M. Tornatore, T. Braun, and S. Giordano, "Measurement and control of geo-location privacy on twitter," *Online social networks and media*, vol. 17, p. 100078, 2020.

[189] R. Aljably, Y. Tian, and M. Al-Rodhaan, "Preserving privacy in multimedia social networks using machine learning anomaly detection," *Security and Communication Networks*, vol. 2020, 2020.

[190] Y. Zhang, M. Humbert, T. Rahman, C.-T. Li, J. Pang, and M. Backes, “Tagvisor: A privacy advisor for sharing hashtags,” in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 287–296.

[191] B. A. AlAhmadi and I. Martinovic, “Malclassifier: Malware family classification using network flow sequence behaviour,” in *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2018, pp. 1–13.

[192] B. Anderson and D. McGrew, “Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity,” in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, 2017, pp. 1723–1732.

[193] H. Shirazi, K. Haefner, and I. Ray, “Fresh-phish: A framework for auto-detection of phishing websites,” in *2017 IEEE international conference on information reuse and integration (IRI)*. IEEE, 2017, pp. 137–143.

[194] B. Cakir and E. Dogdu, “Malware classification using deep learning methods,” in *Proceedings of the ACMSE 2018 Conference*, 2018, pp. 1–5.

[195] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G.-G. Geng, “Boosting the phishing detection performance by semantic analysis,” in *2017 ieee international conference on big data (big data)*. IEEE, 2017, pp. 1063–1070.

[196] H. Shirazi, B. Bezawada, and I. Ray, “” kn0w thy doma1n name” unbiased phishing detection using domain name based features,” in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, 2018, pp. 69–75.

[197] K. Fukuda, J. Heidemann, and A. Qadeer, “Detecting malicious activity with dns backscatter over time,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3203–3218, 2017.

[198] K. Fukushima, T. Nakamura, D. Ikeda, and S. Kiyomoto, “Challenges in classifying privacy policies by machine learning with

word-based features," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 2018, pp. 62–66.

[199] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security.* Springer, 2017, pp. 62–79.

[200] C. Huang, S. Hao, L. Invernizzi, J. Liu, Y. Fang, C. Kruegel, and G. Vigna, "Gossip: Automatically identifying malicious domains from mailing list discussions," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 494–505.

[201] M. Rajab, "An anti-phishing method based on feature analysis," in *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*, 2018, pp. 133–139.

[202] S. Joshi, H. Upadhyay, L. Lagos, N. S. Akkipeddi, and V. Guerra, "Machine learning approach for malware detection using random forest classifier on process list data structure," in *Proceedings of the 2nd International Conference on Information System and Data Mining*, 2018, pp. 98–102.

[203] M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS).* IEEE, 2018, pp. 1–5.

[204] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, 2018, pp. 81–85.

[205] B. A. O. Ikram, B. Mohammed, B. A. Abdelhakim, E. A. Lotfi, and B. Zafar, "Machine learning application for malwares classification using visualization technique," in *Proceedings of the 4th International Conference on Smart City Applications*, 2019, pp. 1–6.

[206] B. Jung, T. Kim, and E. G. Im, "Malware classification using byte sequence information," in *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, 2018, pp. 143–148.

[207] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 3–18.

[208] Y. Mourtaji, M. Bouhorma, and D. Alghazzawi, "Intelligent framework for malware detection with convolutional neural network," in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, 2019, pp. 1–6.

[209] S.-Y. Kung, T. Chanyaswad, J. M. Chang, and P. Wu, "Collaborative pca/dca learning methods for compressive privacy," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, pp. 1–18, 2017.

[210] E. Raff and C. Nicholas, "Malware classification and class imbalance via stochastic hashed lzjd," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 111–120.

[211] D. Liu, Z. Li, K. Du, H. Wang, B. Liu, and H. Duan, "Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 537–552.

[212] H. Sayadi, Y. Gao, H. Mohammadi Makrani, T. Mohsenin, A. Sasan, S. Rafatirad, J. Lin, and H. Homayoun, "Stealthminer: Specialized time series machine learning for run-time stealthy malware detection based on microarchitectural features," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 175–180.

[213] C. Liu, L. Wang, B. Lang, and Y. Zhou, "Finding effective classifier for malicious url detection," in *Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences*, 2018, pp. 240–244.

[214] P. Skolka, C.-A. Staicu, and M. Pradel, "Anything to hide? studying minified and obfuscated code in the web," in *The World Wide Web Conference*, 2019, pp. 1735–1746.

[215] Z. Ren, G. Chen, and W. Lu, "Space filling curve mapping for malware detection and classification," in *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, 2020, pp. 176–180.

[216] P. Shan, Q. Li, P. Zhang, and Y. Gu, "Malware detection method based on control flow analysis," in *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*, 2019, pp. 158–164.

[217] T. Feng and C. Yue, "Visualizing and interpreting rnn models in url-based phishing detection," in *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*, 2020, pp. 13–24.

[218] V. Bannihatti Kumar, R. Iyengar, N. Nisal, Y. Feng, H. Habib, P. Story, S. Cherivirala, M. Hagan, L. Cranor, S. Wilson *et al.*, "Finding a choice in a haystack: Automatic extraction of opt-out statements from privacy policy text," in *Proceedings of The Web Conference 2020*, 2020, pp. 1943–1954.

[219] F. Cozza, A. Guarino, F. Isernia, D. Malandrino, A. Rapuano, R. Schiavone, and R. Zaccagnino, "Hybrid and lightweight detection of third party tracking: Design, implementation, and evaluation," *Computer Networks*, vol. 167, p. 106993, 2020.

[220] W. B. Tesfay, P. Hofmann, T. Nakamura, S. Kiyomoto, and J. Serna, "I read but don't agree: Privacy policy benchmarking using machine learning and the eu gdpr," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 163–166.

[221] ——, “Privacyguide: towards an implementation of the eu gdpr on internet privacy policy evaluation,” in *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*, 2018, pp. 15–21.

[222] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, “Needle in a haystack: Tracking down elite phishing domains in the wild,” in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 429–442.

[223] D. Truong, D. Tran, L. Nguyen, H. Mac, H. A. Tran, and T. Bui, “Detecting web attacks using stacked denoising autoencoder and ensemble learning methods,” in *Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019, pp. 267–272.

[224] Z. Wilkins and N. Zincir-Heywood, “Cougar: clustering of unknown malware using genetic algorithm routines,” in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 1195–1203.

[225] S. Wilson, F. Schaub, F. Liu, K. M. Sathyendra, D. Smullen, S. Zimmeck, R. Ramanath, P. Story, F. Liu, N. Sadeh *et al.*, “Analyzing privacy policies at scale: From crowdsourcing to automated annotations,” *ACM Transactions on the Web (TWEB)*, vol. 13, no. 1, pp. 1–29, 2018.

[226] C. Wu, J. Shi, Y. Yang, and W. Li, “Enhancing machine learning based malware detection model by reinforcement learning,” in *Proceedings of the 8th International Conference on Communication and Network Security*, 2018, pp. 74–78.

[227] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, “Malware analysis of imaged binary samples by convolutional neural network with attention mechanism,” in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, 2018, pp. 127–134.

[228] Y. Huang, Q. Yang, J. Qin, and W. Wen, “Phishing url detection via cnn and attention-based hierarchical rnn,” in *2019*

18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2019, pp. 112–119.

[229] F. Copty, M. Danos, O. Edelstein, C. Eisner, D. Murik, and B. Zeltser, "Accurate malware detection by extreme abstraction," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 101–111.

[230] D. R. Pinto, J. C. Duarte, and R. Sant'Ana, "A deep learning approach to the malware classification problem using autoencoders," in *Proceedings of the XV Brazilian Symposium on Information Systems*, 2019, pp. 1–8.

[231] Z. Sun, Z. Rao, J. Chen, R. Xu, D. He, H. Yang, and J. Liu, "An opcode sequences analysis method for unknown malware detection," in *Proceedings of the 2019 2nd international conference on geoinformatics and data analysis*, 2019, pp. 15–19.

[232] G. McGraw and G. Morrisett, "Attacking malicious code: A report to the infosec research council," *IEEE software*, vol. 17, no. 5, pp. 33–41, 2000.

[233] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: detecting malicious apps in official and alternative android markets." in *NDSS*, vol. 25, no. 4, 2012, pp. 50–52.

[234] A. Bhardwaj, V. Avasthi, H. Sastry, and G. Subrahmanyam, "Ransomware digital extortion: a rising new age threat," *Indian Journal of Science and Technology*, vol. 9, no. 14, pp. 1–5, 2016.

[235] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems.* Springer, 2002, pp. 251–260.

[236] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Third international symposium on information processing in sensor networks, 2004. IPSN 2004.* IEEE, 2004, pp. 259–268.

[237] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 243–258.

[238] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 551–560.

[239] A. Mukherjee and J. McGinnis, "E-healthcare: an analysis of key themes in research," *International Journal of Pharmaceutical and Healthcare Marketing*, 2007.

[240] O. Boric-Lubecke, X. Gao, E. Yavari, M. Baboli, A. Singh, and V. M. Lubecke, "E-healthcare: Remote monitoring, privacy, and security," in *2014 IEEE MTT-S International Microwave Symposium (IMS2014)*. IEEE, 2014, pp. 1–3.

[241] M. Lippi, P. Pałka, G. Contissa, F. Lagioia, H.-W. Micklitz, G. Sartor, and P. Torroni, "Claudette: an automated detector of potentially unfair clauses in online terms of service," *Artificial Intelligence and Law*, vol. 27, no. 2, pp. 117–139, 2019.

[242] M. Héder, "From nasa to eu: the evolution of the trl scale in public sector innovation," *The Innovation Journal*, vol. 22, no. 2, pp. 1–23, 2017.

[243] D. Malandrino and V. Scarano, "Privacy leakage on the Web: Diffusion and countermeasures," *Computer Networks*, vol. 57, no. 14, pp. 2833 – 2855, 2013.

[244] "Interactive Advertising Bureau (IAB) and PricewaterhouseCoopers (PwC) US. Internet Advertising Revenue Report," https://www.iab.com/news/iab-advertising-revenue-q1-2019/, 2019.

[245] D. Malandrino and V. Scarano, "Supportive, Comprehensive and Improved Privacy Protection for Web Browsing," in *Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and Third International Conference on Social Computing (SocialCom), Boston, MA, USA*, 2011, pp. 1173–1176.

[246] R. Masood, B. Z. H. Zhao, H. J. Asghar, and M. A. Kâafar, "Touch and You're Trapp(ck)ed: Quantifying the Uniqueness of Touch Gestures for Tracking," *PoPETs*, vol. 2018, no. 2, pp. 122–142, 2018.

[247] Y. He, B. Hu, and Z. Han, "Dynamic Privacy Leakage Analysis of Android Third-Party Libraries," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 2018, pp. 275–280.

[248] R. Binns, U. Lyngs, M. Van Kleek, J. Zhao, T. Libert, and N. Shadbolt, "Third Party Tracking in the Mobile Ecosystem," in *Proceedings of the 10th ACM Conference on Web Science*, ser. WebSci '18, 2018, pp. 23–31.

[249] B. Krishnamurthy, D. Malandrino, and C. E. Wills, "Measuring privacy loss and the impact of privacy protection in web browsing," in *Proc. of the 3rd Symposium on Usable Privacy and Security*, ser. SOUPS '07, 2007, pp. 52–63.

[250] B. Krishnamurthy and C. Wills, "Privacy Diffusion on the Web: A Longitudinal Perspective," in *Proc. of the 18th Int. Conf. on World Wide Web*, 2009, pp. 541–550.

[251] D. Malandrino, A. Petta, V. Scarano, L. Serra, R. Spinelli, and B. Krishnamurthy, "Privacy Awareness About Information Leakage: Who Knows What About Me?" in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, ser. WPES '13, 2013, pp. 279–284.

[252] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, Oct. 2002.

[253] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, ser. SP '08, 2008, pp. 111–125.

[254] R. Gross and A. Acquisti, "Information Revelation and Privacy in Online Social Networks," in *Proc. of the 2005 ACM Workshop on Privacy in the Electronic Society*, 2005, pp. 71–80.

[255] C. Dwyer, “Privacy in the Age of Google and Facebook,” *IEEE Technology and Society Magazine*, vol. 30, no. 3, pp. 58–63, 2011.

[256] D. Perito, C. Castelluccia, M. Kaafar, and P. Manils, “How Unique and Traceable Are Usernames?” in *In Proc. of the 11th Int. Conf. on Privacy Enhancing Technologies*, 2011, vol. 6794, pp. 1–17.

[257] H. Kim and J. Huh, “Perceived Relevance and Privacy Concern Regarding Online Behavioral Advertising (OBA) and Their Role in Consumer Responses,” *Journal of Current Issues & Research in Advertising*, vol. 38, no. 1, pp. 92–105, 2017.

[258] L. F. Cranor, “Can Users Control Online Behavioral Advertising Effectively?” *IEEE Security Privacy*, vol. 10, no. 2, pp. 93–96, 2012.

[259] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, “Cookies That Give You Away: The Surveillance Implications of Web Tracking,” in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW ’15, 2015, pp. 289–299.

[260] C. E. Wills and D. C. Uzunoglu, “What Ad Blockers Are (and Are Not) Doing,” in *Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2016, Washington, DC, USA, October 24-25, 2016*, 2016, pp. 72–77.

[261] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, “Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools,” in *2017 IEEE European Symposium on Security and Privacy*, 2017, pp. 319–333.

[262] P. Leon, B. Ur, R. Shay, Y. Wang, R. Balebako, and L. Cranor, “Why Johnny Can’t Opt Out: A Usability Evaluation of Tools to Limit Online Behavioral Advertising,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 589–598.

[263] “Allowing acceptable ads in adblock plus - agreements,” https://adblockplus.org/acceptable-ads-agreements, accessed on June 28th, 2019.

[264] S. D'Ambrosio, S. D. Pasquale, G. Iannone, D. Malandrino, A. Negro, G. Patimo, A. Petta, V. Scarano, L. Serra, and R. Spinelli, "Mobile phone batteries draining: Is green web browsing the solution?" in *International Green Computing Conference, IGCC 2014, Dallas, TX, USA, November 3-5, 2014*, 2014, pp. 1–10.

[265] S. D'Ambrosio, S. D. Pasquale, G. Iannone, D. Malandrino, A. Negro, G. Patimo, V. Scarano, R. Spinelli, and R. Zaccagnino, "Privacy as a proxy for Green Web browsing: Methodology and experimentation," *Computer Networks*, vol. 126, pp. 81–99, 2017.

[266] D. Gugelmann, M. Happe, B. Ager, and V. Lenders, "An Automated Approach for Complementing Ad Blockers' Blacklists," *PoPETs*, vol. 2015, no. 2, pp. 282–298, 2015.

[267] T.-C. Li, H. Hang, M. Faloutsos, and P. Efstathopoulos, "TrackAdvisor: Taking Back Browsing Privacy from Third-Party Trackers," in *Passive and Active Measurement*, J. Mirkovic and Y. Liu, Eds. Cham: Springer International Publishing, 2015, pp. 277–289.

[268] M. Ikram, H. J. Asghar, M. A. Kâafar, A. Mahanti, and B. Krishnamurthy, "Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning," *PoPETs*, vol. 2017, no. 1, pp. 79–99, 2017.

[269] Q. Wu, Q. Liu, Y. Zhang, and G. Wen, "TrackerDetector: A system to detect third-party trackers through machine learning," *Computer Networks*, vol. 91, pp. 164 – 173, 2015.

[270] B. Krishnamurthy, K. Naryshkin, and C. E. Wills, "Privacy leakage vs. Protection measures: the growing disconnect," in *Web 2.0 Security and Privacy Workshop*, 2011, http://www2.research.att.com/~bala/papers/w2sp11.pdf.

[271] M. Tran, X. Dong, Z. Liang, and X. Jiang, "Tracking the Trackers: Fast and Scalable Dynamic Analysis of Web Content for Privacy Violations," in *Applied Cryptography and Network Security*, F. Bao, P. Samarati, and J. Zhou, Eds., 2012, pp. 418–435.

[272] B. Krishnamurthy and C. E. Wills, “On the leakage of personally identifiable information via online social networks,” in *Proceedings of the 2nd ACM workshop on Online social networks*, ser. WOSN ’09, 2009, pp. 7–12.

[273] C. J. Bennett, “Cookies, web bugs, webcams and cue cats: Patterns of surveillance on the world wide web,” *Ethics and Information Technology*, vol. 3, no. 3, pp. 195–208, 2001.

[274] D. Martin, H. Wu, and A. Alsaid, “Hidden Surveillance by Web Sites: Web Bugs in Contemporary Use,” *Commun. ACM*, vol. 46, no. 12, pp. 258–264, Dec. 2003.

[275] D. Jang, R. Jhala, S. Lerner, and H. Shacham, “An empirical study of privacy-violating information flows in JavaScript web applications,” in *Proceedings of the 17th ACM conference on Computer and communications security.* ACM, 2010, pp. 270–283.

[276] L. Olejnik, C. Castelluccia, and A. Janc, “Why johnny can’t browse in peace: On the uniqueness of web browsing history patterns,” in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, 2012.

[277] W. Palant, “AdBlock Plus,” http://adblockplus.org/, Accessed on June 29, 2018.

[278] “AdBlock Plus Filters Explained,” https://adblockplus.org/filter-cheatsheet, Accessed on June 28th, 2019.

[279] Ghostery, https://www.ghostery.com, Accessed on June 28th, 2019.

[280] Disconnect, https://disconnect.me, Accessed on June 28th, 2019.

[281] NoScriptLite, https://mybrowseraddon.com/noscript-lite.html, accessed on June 28th, 2019.

[282] PrivacyBadger, https://www.eff.org/privacybadger, Accessed on June 28th, 2019.

[283] A. Gervais, A. Filios, V. Lenders, and S. Capkun, "Quantifying Web Adblocker Privacy," in *ESORICS (2)*, ser. Lecture Notes in Computer Science, vol. 10493. Springer, 2017, pp. 21–42.

[284] K. Rzecki, P. PÃÂawiak, M. NiedÃÂºwiecki, T. SoÃÂnicki, J. LeÃÂkow, and M. Ciesielski, "Person recognition based on touch screen gestures using computational intelligence methods," *Information Sciences*, vol. 415-416, pp. 70 – 84, 2017.

[285] A. Cosimato, R. De Prisco, A. Guarino, N. Lettieri, D. Malandrino, G. Sorrentino, and R. Zaccagnino, "The conundrum of success in music: playing it or talking about it?" *IEEE Access*, pp. 1–10, 2019.

[286] F. Deeba, S. K. Mohammed, F. M. Bui, and K. A. Wahid, "Learning from imbalanced data: A comprehensive comparison of classifier performance for bleeding detection in endoscopic video," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2016, pp. 1006–1009.

[287] M. E. Maron, "Automatic indexing: an experimental inquiry," *Journal of the ACM (JACM)*, vol. 8, no. 3, pp. 404–417, 1961.

[288] J. R. Quinlan, *C4. 5: programs for machine learning.* Elsevier, 2014.

[289] S. Avasarala, *Selenium WebDriver practical guide.* Packt Publishing Ltd, 2014.

[290] E. Lawrence, *Debugging with Fiddler: The complete reference from the creator of the Fiddler Web Debugger.* Eric Lawrence, 2012.

[291] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit.* Elsevier, 2006.

[292] T.-C. Li, H. Hang, M. Faloutsos, and P. Efstathopoulos, "Trackadvisor: Taking back browsing privacy from third-party trackers," in *International Conference on Passive and Active Network Measurement.* Springer, 2015, pp. 277–289.

[293] D. Malandrino and V. Scarano, "Privacy leakage on the web: Diffusion and countermeasures," *Computer Networks*, vol. 57, no. 14, pp. 2833–2855, 2013.

[294] M. Lai, "On language and structure in polarized communities," Ph.D. dissertation, 2019.

[295] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[296] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* Springer, 2013, vol. 112.

[297] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "The rise of panopticons: Examining region-specific third-party web tracking," in *International Workshop on Traffic Monitoring and Analysis.* Springer, 2014, pp. 104–114.

[298] A. Guarino, N. Lettieri, D. Malandrino, P. Russo, and R. Zaccagnino, "Visual analytics to make sense of large-scale administrative and normative data," in *2019 23rd International Conference Information Visualisation (IV).* IEEE, 2019, pp. 133–138.

[299] N. Lettieri, A. Guarino, and D. Malandrino, "E-science and the law. three experimental platforms for legal analytics." in *JURIX*, 2018, pp. 71–80.

[300] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler, "Visual analytics: Scope and challenges," in *Visual data mining.* Springer, 2008, pp. 76–90.

[301] C. Wang, "Graph-based techniques for visual analytics of scientific data sets," *Computing in Science & Engineering*, vol. 20, no. 1, pp. 93–103, 2018.

[302] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[303] T. Saito, T. Noda, R. Hosoya, K. Tanabe, and Y. Saito, "On Estimating Platforms of Web User with JavaScript Math Object," in *International Conference on Network-Based Information Systems.* Springer, 2018, pp. 407–418.

[304] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242. Piscataway, NJ, 2003, pp. 133–142.

[305] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[306] J. Hoak, "The effects of outliers on support vector machines," *Portland State University*, 2010.

[307] M. Debruyne *et al.*, "An outlier map for support vector machine classification," *The Annals of Applied Statistics*, vol. 3, no. 4, pp. 1566–1580, 2009.

[308] T. Kanamori, S. Fujiwara, and A. Takeda, "Breakdown point of robust support vector machine," *arXiv preprint arXiv:1409.0934*, 2014.

[309] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, January 1967.

[310] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery?: analyzing mobile browser energy consumption," ser. WWW'12, 2012, pp. 41–50.

[311] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, "Cookies that give you away: The surveillance implications of web tracking," in *Proceedings of the 24th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2015, pp. 289–299.

[312] J. Mayer, “Tracking the Trackers: Self-Help Tools,” https://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools, September 13, 2011.

[313] T. Barbaro, Michael; Zeller Jr, “A Face Is Exposed for AOL Searcher No. 4417749,” 2006, the New York Times.

[314] A. Narayanan and V. Shmatikov, “De-anonymizing Social Networks,” in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, ser. SP ’09, 2009, pp. 173–187.

[315] B. Krishnamurthy and J. Rexford, *Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement.* Addison-Wesley, 2001.

[316] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, “Cookieless monster: Exploring the ecosystem of web-based device fingerprinting,” in *Security and privacy (SP), 2013 IEEE symposium on.* IEEE, 2013.

[317] C. E. Wills and D. C. Uzunoglu, “What Ad Blockers Are (and Are Not) Doing,” in *Hot Topics in Web Systems and Technologies (HotWeb), 2016 Fourth IEEE Workshop on.* IEEE, 2016.

[318] N. Kushmerick, “Learning to Remove Internet Advertisements,” in *Proceedings of the Third Annual Conference on Autonomous Agents*, ser. AGENTS ’99, 1999, pp. 175–181.

[319] C. R. Orr, A. Chauhan, M. Gupta, C. J. Frisz, and C. W. Dunn, “An Approach for Identifying JavaScript-loaded Advertisements Through Static Program Analysis,” in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, ser. WPES ’12, 2012.

[320] S. Bhagavatula, C. Dunn, C. Kanich, M. Gupta, and B. Ziebart, “Leveraging Machine Learning to Improve Unwanted Resource Filtering,” in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, ser. AISec, 2014.

[321] H. Haddadi, P. Hui, and I. Brown, “Mobiad: private and scalable mobile advertising,” in *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*. ACM, 2010, pp. 33–38.

[322] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, “Adnostic: Privacy preserving targeted advertising,” in *Proceedings Network and Distributed System Symposium*, 2010.

[323] S. Guha, B. Cheng, and P. Francis, “Privad: Practical privacy in online advertising,” in *USENIX conference on Networked systems design and implementation*, 2011, pp. 169–182.

[324] Y. Liu and A. Simpson, “Privacy-preserving targeted mobile advertising: Formal models and analysis,” in *Data Privacy Management and Security Assurance*. Springer, 2016, pp. 94–110.

[325] J. Parra-Arnau, J. P. Achara, and C. Castelluccia, “Myadchoices: Bringing transparency and control to online advertising,” *ACM Transactions on the Web (TWEB)*, vol. 11, no. 1, p. 7, 2017.

[326] J. Parra-Arnau, “Pay-per-tracking: A collaborative masking model for web browsing,” *Information Sciences*, vol. 385, pp. 96–124, 2017.

[327] D. Sánchez and A. Viejo, “Privacy-preserving and advertising-friendly web surfing,” *Computer Communications*, vol. 130, pp. 113–123, 2018.

[328] A. Yamada, H. Masanori, and Y. Miyake, “Web Tracking Site Detection Based on Temporal Link Analysis,” in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, 2010.

[329] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, “Building text classifiers using positive and unlabeled examples,” in *Third IEEE International Conference on Data Mining*, Nov 2003.

[330] C. Elkan and K. Noto, “Learning Classifiers from Only Positive and Unlabeled Data,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[331] G. Beigi, R. Guo, A. Nou, Y. Zhang, and H. Liu, "Protecting user privacy: An approach for untraceable web browsing history and unambiguous user profiles," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 2019, pp. 213–221.

[332] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[333] A. Cuzzocrea, F. Martinelli, and F. Mercaldo, "A machine-learning framework for supporting intelligent web-phishing detection and analysis," in *Proceedings of the 23rd International Database Applications & Engineering Symposium*. ACM, 2019, p. 43.

[334] M. Romanelli, C. Palamidessi, and K. Chatzikokolakis, "Generating optimal privacy-protection mechanisms via machine learning," *arXiv preprint arXiv:1904.01059*, 2019.

[335] "Where Do People Spend the Most Time on Social Media?" https://www.statista.com/chart/18983/time-spent-on-social-media/, Accessed on July 6th, 2020.

[336] Y. Wang, G. Norcie, S. Komanduri, A. Acquisti, P. G. Leon, and L. F. Cranor, "" i regretted the minute i pressed share" a qualitative study of regrets on facebook," in *Proceedings of the seventh symposium on usable privacy and security*, 2011, pp. 1–16.

[337] M. Sleeper, J. Cranshaw, P. G. Kelley, B. Ur, A. Acquisti, L. F. Cranor, and N. Sadeh, "" i read my twitter the next morning and was astonished" a conversational perspective on twitter regrets," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2013, pp. 3277–3286.

[338] Y. Yang, J. Lutes, F. Li, B. Luo, and P. Liu, "Stalking online: on user privacy in social networks," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, 2012, pp. 37–48.

[339] D. Lazer, A. S. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann *et al.*, “Life in the network: the coming age of computational social science,” *Science (New York, NY)*, vol. 323, no. 5915, p. 721, 2009.

[340] “Number of compromised data records in selected data breaches as of April 2020.”

[341] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, “Cookies that give you away: The surveillance implications of web tracking,” in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 289–299.

[342] “Adblock,” http://adblock.mozdev.org/, Accessed on June 29, 2018.

[343] M. Ikram, H. J. Asghar, M. A. Kaafar, A. Mahanti, and B. Krishnamurthy, “Towards seamless tracking-free web: Improved detection of trackers via one-class learning,” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 79–99, 2017.

[344] P. N. Hiremath, J. Armentrout, S. Vu, T. N. Nguyen, Q. T. Minh, and P. H. Phung, “Mywebguard: Toward a user-oriented tool for security and privacy protection on the web,” in *International Conference on Future Data and Security Engineering.* Springer, 2019, pp. 506–525.

[345] Z. Cai, Z. He, X. Guan, and Y. Li, “Collective data-sanitization for preventing sensitive information inference attacks in social networks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2016.

[346] T. C. Leonard, “Richard h. thaler, cass r. sunstein, nudge: Improving decisions about health, wealth, and happiness,” 2008.

[347] R. Spence, *Information visualization.* Springer, 2001, vol. 1.

[348] C. Ware, *Information visualization: perception for design.* Morgan Kaufmann, 2019.

[349] S. dâAmbrosio, S. De Pasquale, G. Iannone, D. Malandrino, A. Negro, G. Patimo, V. Scarano, R. Spinelli, and R. Zaccagnino, "Privacy as a proxy for green web browsing: Methodology and experimentation," *Computer Networks*, vol. 126, pp. 81–99, 2017.

[350] A. L. Allen, "Protecting one's own privacy in a big data economy," *Harv. L. Rev. F.*, vol. 130, p. 71, 2016.

[351] T. Bujlow, V. Carela-Español, J. Sole-Pareta, and P. Barlet-Ros, "A survey on web tracking: Mechanisms, implications, and defenses," *Proceedings of the IEEE*, vol. 105, no. 8, pp. 1476–1510, 2017.

[352] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *Journal of Network and Computer Applications*, vol. 126, pp. 45–58, 2019.

[353] B. Krishnamurthy and C. E. Wills, "On the leakage of personally identifiable information via online social networks," in *Proceedings of the 2nd ACM workshop on Online social networks*, 2009, pp. 7–12.

[354] L. Sweeney, "Simple demographics often identify people uniquely," *Health (San Francisco)*, vol. 671, no. 2000, pp. 1–34, 2000.

[355] "US babes names," https://www.kaggle.com/kaggle/us-baby-names, Accessed on July 6th, 2020.

[356] "Census Bureau. Us surnames," https://www.census.gov/topics/population/genealogy/data/2010_surnames.html, Accessed on July 6th, 2020.

[357] "Most populate cities," https://simplemaps.com/data/world-cities, Accessed on July 6th, 2020.

[358] "Manual ICD-9-CM," http://www.salute.gov.it/portale/documentazione/p6_2_2_1.jsp?lingua=italiano&id=2251, Accessed on July 6th, 2020.

[359] H. Mao, X. Shuai, and A. Kapadia, "Loose tweets: an analysis of privacy leaks on twitter," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society.* ACM, 2011, pp. 1–12.

[360] "Election Day Tweets," https://www.kaggle.com/kinguistics/election-day-tweets?select=election_day_tweets.csv, Accessed on July 6th, 2020.

[361] "Medical Transcript," https://www.kaggle.com/tboyle10/medicaltranscriptions, Accessed on July 6th, 2020.

[362] "Medical Speech, Transcription, and Intent," https://www.kaggle.com/paultimothymooney/medical-speech-transcription-and-intent, Accessed on July 6th, 2020.

[363] "Amazon Job skills," https://www.kaggle.com/atahmasb/amazon-job-skills, Accessed on July 6th, 2020.

[364] "Twitter US Airline Sentiment," https://www.kaggle.com/crowdflower/twitter-airline-sentiment, Accessed on July 6th, 2020.

[365] "The Movies dataset," https://www.kaggle.com/rounakbanik/the-movies-dataset, Accessed on July 6th, 2020.

[366] P. Koehn and J. Schroeder, "Experiments in domain adaptation for statistical machine translation," in *Proceedings of the second workshop on statistical machine translation*, 2007, pp. 224–227.

[367] M. L. McHugh, "Interrater reliability: the kappa statistic," *Biochemia medica*, vol. 22, no. 3, pp. 276–282, 2012.

[368] N. S. Tawfik and M. R. Spruit, "Towards recognition of textual entailment in the biomedical domain," in *International Conference on Applications of Natural Language to Information Systems.* Springer, 2019, pp. 368–375.

[369] T.-Y. Chang and Y.-N. Chen, "What does this word mean? explaining contextualized embeddings with natural language definition," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 6066–6072.

[370] H. Chen, S. McKeever, and S. J. Delany, "The use of deep learning distributed representations in the identification of abusive text," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, 2019, pp. 125–133.

[371] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 8018–8025.

[372] A. Panda, A. Gonawela, S. Acharyya, D. Mishra, M. Mohapatra, R. Chandrasekaran, and J. Pal, "Nivaduck-a scalable pipeline to build a database of political twitter handles for india and the united states," in *International Conference on Social Media and Society*, 2020, pp. 200–209.

[373] R. Zhao, W. B. Haskell, and V. Y. Tan, "Stochastic l-bfgs: Improved convergence rates and practical acceleration strategies," *IEEE Transactions on Signal Processing*, vol. 66, no. 5, pp. 1155–1169, 2018.

[374] S. S. SHAPIRO and M. B. WILK, "An analysis of variance test for normality (complete samples)â ," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965.

[375] P. E. McKight and J. Najab, "Kruskal-wallis test," *The corsini encyclopedia of psychology*, pp. 1–1, 2010.

[376] J. M. Rumbold and B. K. Pierscionek, "What are data? a categorization of the data sensitivity spectrum," *Big data research*, vol. 12, pp. 49–59, 2018.

[377] L. Calderoni, M. Ferrara, A. Franco, and D. Maio, "Indoor localization in a hospital environment using random forest classifiers,"

Expert Systems with Applications, vol. 42, no. 1, pp. 125–134, 2015.

[378] K. Moore and J. C. McElroy, "The influence of personality on facebook usage, wall postings, and regret," *Computers in human behavior*, vol. 28, no. 1, pp. 267–274, 2012.

[379] Y. Chen, W. Yan, C. Li, Y. Huang, and L. Yang, "Personalized optimal bicycle trip planning based on q-learning algorithm," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[380] Z. Cheng, N. Chen, B. Liu, Z. Gao, L. Huang, X. Du, and M. Guizani, "Joint user association and resource allocation in hetnets based on user mobility prediction," *Computer Networks*, p. 107312, 2020.

[381] W. Liu, L. Wang, E. Wang, Y. Yang, D. Zeghlache, and D. Zhang, "Reinforcement learning-based cell selection in sparse mobile crowdsensing," *Computer Networks*, vol. 161, pp. 102–114, 2019.

[382] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167–176.

[383] D. Aberdeen, O. Pacovsky, and A. Slater, "The learning behind gmail priority inbox," 2010.

[384] Y. Koren, E. Liberty, Y. Maarek, and R. Sandler, "Automatically tagging email by leveraging other users' folders," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 913–921.

[385] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, no. Mar, pp. 551–585, 2006.

[386] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[387] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[388] Y. Li and A. Ngom, "Data integration in machine learning," in *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2015, pp. 1665–1671.

[389] Y. Li, F.-X. Wu, and A. Ngom, "A review on machine learning principles for multi-view biological data integration," *Briefings in bioinformatics*, vol. 19, no. 2, pp. 325–340, 2018.

[390] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[391] A. Caliskan Islam, J. Walsh, and R. Greenstadt, "Privacy detective: Detecting private information and collective privacy behavior in a large social network," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 35–46.

[392] R. De Prisco, D. Malandrino, D. Pirozzi, G. Zaccagnino, and R. Zaccagnino, "Understanding the structure of musical compositions: Is visualization an effective approach?" *Information Visualization*, vol. 16, no. 2, pp. 139–152, 2017.

[393] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[394] J. He, W. W. Chu, and Z. V. Liu, "Inferring privacy information from social networks," in *International Conference on Intelligence and Security Informatics.* Springer, 2006, pp. 154–165.

[395] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation.* Springer, 2008, pp. 1–19.

[396] M. Mondal, J. Messias, S. Ghosh, K. P. Gummadi, and A. Kate, "Forgetting in social media: Understanding and controlling longitudinal exposure of socially shared data," in *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*, 2016, pp. 287–299.

[397] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[398] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *2007 IEEE 23rd International Conference on Data Engineering.* IEEE, 2007, pp. 106–115.

[399] A. Haeberlen, B. C. Pierce, and A. Narayan, “Differential privacy under fire.” in *USENIX Security Symposium*, vol. 33, 2011.

[400] J. H. Abawajy, M. I. H. Ninggal, Z. Al Aghbari, A. B. Darem, and A. Alhashmi, “Privacy threat analysis of mobile social network data publishing,” in *International Conference on Security and Privacy in Communication Systems.* Springer, 2017, pp. 60–68.

[401] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, “Persona: an online social network with user-defined privacy,” in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 135–146.

[402] S. Jahid, P. Mittal, and N. Borisov, “Easier: Encryption-based access control in social networks with efficient revocation,” in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 411–415.

[403] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[404] Q. Wang, J. Bhandal, S. Huang, and B. Luo, “Classification of private tweets using tweet content,” in *2017 IEEE 11th International Conference on Semantic Computing (ICSC).* IEEE, 2017, pp. 65–68.

[405] ——, “Content-based classification of sensitive tweets,” *International Journal of Semantic Computing*, vol. 11, no. 04, pp. 541–562, 2017.

[406] P. Salesses, K. Schechtner, and C. A. Hidalgo, "The collaborative image of the city: mapping the inequality of urban perception," *PloS one*, vol. 8, no. 7, p. e68400, 2013.

[407] PewResearchCenter, "Teens, social media & technology 2018," 2018, https://www.pewinternet.org/2018/05/31/teens-social-media-technology-2018/.

[408] M. C. Buzzi, M. Buzzi, B. Leporini, and A. Trujillo, "Analyzing visually impaired people's touch gestures on smartphones," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 5141–5169, Feb 2017.

[409] K. Rzecki, P. Pławiak, M. Niedźwiecki, T. Sośnicki, J. Leśkow, and M. Ciesielski, "Person recognition based on touch screen gestures using computational intelligence methods," *Information Sciences*, vol. 415, pp. 70–84, 2017.

[410] C. Shen, Y. Chen, and X. Guan, "Performance evaluation of implicit smartphones authentication via sensor-behavior analysis," *Information Sciences*, vol. 430, pp. 538–553, 2018.

[411] G. S. Eskander, R. Sabourin, and E. Granger, "A bio-cryptographic system based on offline signature images," *Information Sciences*, vol. 259, pp. 170–191, 2014.

[412] S. Bernardini, K. Porayska-Pomsta, and T. J. Smith, "Echoes: An intelligent serious game for fostering social communication in children with autism," *Information Sciences*, vol. 264, pp. 41–60, 2014.

[413] A. Wojciechowski and R. Al-Musawi, "Assisstive technology application for enhancing social and language skills of young children with autism," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 5419–5439, Feb 2017.

[414] R. Klempous, J. Nikodem, W. Jacak, and Z. Chaczko, *Advanced Methods and Applications in Computational Intelligence*. Springer Science & Business Media, 2013, vol. 6.

[415] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, “Evaluation of machine learning classifiers for mobile malware detection,” *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.

[416] Y. Li, H. Hu, and G. Zhou, “Using data augmentation in continuous authentication on smartphones,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 628–640, 2018.

[417] Y. Li, H. Hu, Z. Zhu, and G. Zhou, “Scanet: sensor-based continuous authentication with two-stream convolutional neural networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 3, pp. 1–27, 2020.

[418] Y. Li, B. Zou, S. Deng, and G. Zhou, “Using feature fusion strategies in continuous authentication on smartphones,” *IEEE Internet Computing*, vol. 24, no. 2, pp. 49–56, 2020.

[419] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon, “Biometric-rich gestures: a novel approach to authentication on multi-touch devices,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 2012, pp. 977–986.

[420] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, “Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication,” *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 136–148, 2013.

[421] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang, “Silentsense: silent user identification via touch and movement behavioral biometrics,” in *Proceedings of the 19th annual international conference on Mobile computing & networking.* ACM, 2013, pp. 187–190.

[422] R. Masood, B. Z. H. Zhao, H. J. Asghar, and M. A. Kaafar, “Touch and you’re trapp (ck) ed: Quantifying the uniqueness of touch gestures for tracking,” *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 2, pp. 122–142, 2018.

[423] V. Nacher, J. Jaen, E. Navarro, A. Catala, and P. González, "Multi-touch gestures for pre-kindergarten children," *International Journal of Human-Computer Studies*, vol. 73, pp. 37–51, 2015.

[424] C. Villamor, D. Willis, and L. Wroblewski, "Touch gesture reference guide," *Touch Gesture Reference Guide*, 2010.

[425] L. Ballard, D. Lopresti, and F. Monrose, "Evaluating the security of handwriting biometrics," in *Tenth International Workshop on Frontiers in Handwriting Recognition.* Suvisoft, 2006.

[426] R.-D. Vatavu, L. Anthony, and Q. Brown, "Child or adult? Inferring Smartphone users' age group from touch measurements alone," in *IFIP Conference on Human-Computer Interaction.* Springer, 2015, pp. 1–9.

[427] L. Anthony, Q. Brown, B. Tate, J. Nias, R. Brewer, and G. Irwin, "Designing smarter touch-based interfaces for educational contexts," *Personal and Ubiquitous Computing*, vol. 18, no. 6, pp. 1471–1483, 2014.

[428] Toan, Nguyen and Aditi, Roy and Nasir, Memon.

[429] B. J. Choo, H.-J. Joo, B.-S. Kim, S.-h. Cha, and D.-H. Myung, "Using multimedia based big data for child abuse prevention system (based on the establishment of e-child welfare support system)," *Multimedia Tools and Applications*, vol. 78, no. 20, pp. 28 805–28 814, Oct 2019.

[430] S. Livingstone and L. Haddon, "Eu kids online." *Zeitschrift Für Psychologie/Journal of Psychology*, vol. 217, no. 4, p. 236, 2009.

[431] K. J. Mitchell, D. Finkelhor, and J. Wolak, "Protecting youth online: Family use of filtering and blocking software," *Child abuse & neglect*, vol. 29, no. 7, pp. 753–765, 2005.

[432] C. R. Richardson, P. J. Resnick, D. L. Hansen, H. A. Derry, and V. J. Rideout, "Does pornography-blocking software block access to health information on the internet?" *JAMA*, vol. 288, no. 22, pp. 2887–2894, 2002.

[433] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[434] Y. Li, F.-X. Wu, and A. Ngom, "A review on machine learning principles for multi-view biological data integration," *Briefings in bioinformatics*, vol. 19, no. 2, pp. 325–340, 2016.

[435] P. Pavlidis, J. Weston, J. Cai, and W. S. Noble, "Learning gene functional classifications from multiple data types," *Journal of computational biology*, vol. 9, no. 2, pp. 401–411, 2002.

[436] M. Fratello, G. Caiazzo, F. Trojsi, A. Russo, G. Tedeschi, R. Tagliaferri, and F. Esposito, "Multi-view ensemble classification of brain connectivity images for neurodegeneration type discrimination," *Neuroinformatics*, vol. 15, no. 2, pp. 199–213, 2017.

[437] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[438] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *24th international conference on Machine learning*, 2007, pp. 935–942.

[439] A. Agrawal and T. Menzies, "Is better data better than better data miners?: on the benefits of tuning smote for defect prediction," in *Proceedings of the 40th International Conference on Software Engineering.* ACM, 2018, pp. 1050–1061.

[440] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863–905, 2018.

[441] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[442] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[443] T. Fawcett, “An introduction to ROC analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[444] M. Hossin and M. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[445] J. A. Swets, “Measuring the accuracy of diagnostic systems,” *Science*, vol. 240, no. 4857, pp. 1285–1293, 1988.

[446] J. A. Obar and A. Oeldorf-Hirsch, “The biggest lie on the internet: Ignoring the privacy policies and terms of service policies of social networking services,” *Information, Communication & Society*, vol. 23, no. 1, pp. 128–147, 2020.

[447] A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T. B. Norton, N. C. Russell, P. Story, J. Reidenberg, and N. Sadeh, “Privonto: A semantic framework for the analysis of privacy policies,” *Semantic Web*, vol. 9, no. 2, pp. 185–203, 2018.

[448] A. M. McDonald and L. F. Cranor, “The cost of reading privacy policies,” *Isjlp*, vol. 4, p. 543, 2008.

[449] J. R. Reidenberg, N. C. Russell, A. J. Callen, S. Qasir, and T. B. Norton, “Privacy harms and the effectiveness of the notice and choice framework,” *ISJLP*, vol. 11, p. 485, 2015.

[450] H.-W. Micklitz, P. Pałka, and Y. Panagis, “The empire strikes back: digital control of unfair terms of online services,” *Journal of consumer policy*, vol. 40, no. 3, pp. 367–388, 2017.

[451] Z. S. Jalali, W. Wang, M. Kim, H. Raghavan, and S. Soundarajan, “On the information unfairness of social networks,” in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 613–521.

[452] K. Bakour and H. M. Ünver, “Visdroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques,” *Neural Computing and Applications*, pp. 1–21, 2020.

[453] R. S. Rao and A. R. Pais, “Detection of phishing websites using an efficient feature-based machine learning framework,” *Neural Computing and Applications*, vol. 31, no. 8, pp. 3851–3873, 2019.

[454] M. Loos and J. Luzak, “Wanted: a bigger stick. on unfair terms in consumer contracts with online service providers,” *Journal of consumer policy*, vol. 39, no. 1, pp. 63–90, 2016.

[455] W. House, “Big data: seizing opportunities, preserving values (report for the president),” *Washington DC, USA: Executive Office of the President.[WWW document] https://bit.ly/31VESSF*, 2014.

[456] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung *et al.*, “Multilingual universal sentence encoder for semantic retrieval,” *arXiv preprint arXiv:1907.04307*, 2019.

[457] G. Nayak, R. Ghosh, X. Jia, V. Mithafi, and V. Kumar, “Semi-supervised classification using attention-based regularization on coarse-resolution data,” in *Proceedings of the 2020 SIAM International Conference on Data Mining.* SIAM, 2020, pp. 253–261.

[458] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction.* John Wiley & Sons, 2010.

[459] R. Andreoli, A. Corolla, A. Faggiano, D. Malandrino, D. Pirozzi, M. Ranaldi, G. Santangelo, and V. Scarano, “A framework to design, develop, and evaluate immersive and collaborative serious games in cultural heritage,” *ACM Journal on Computing and Cultural Heritage*, vol. 11, no. 1, pp. 4:1–4:22, 2018.

[460] R. De Prisco, D. Malandrino, D. Pirozzi, G. Zaccagnino, and R. Zaccagnino, “Understanding the structure of musical compositions: Is visualization an effective approach?” *Inf. Vis.*, vol. 16, no. 2, pp. 139–152, 2017.

[461] P. G. Leon, B. Ur, R. Shay, Y. Wang, R. Balebako, and L. F. Cranor, "Why johnny can't opt out: a usability evaluation of tools to limit online behavioral advertising," in *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*, J. A. Konstan, E. H. Chi, and K. Höök, Eds. ACM, 2012, pp. 589–598.

[462] N. Lettieri, A. Altamura, and D. Malandrino, "The legal macroscope: Experimenting with visual legal analytics," *Inf. Vis.*, vol. 16, no. 4, pp. 332–345, 2017.

[463] R. A. Virzi, "Refining the test phase of usability evaluation: How many subjects is enough?" *Human Factors*, vol. 34, no. 4, pp. 457–468, 1992.

[464] J. Nielsen and T. K. Landauer, "A Mathematical Model of the Finding of Usability Problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, 1993, pp. 206–213.

[465] J. P. Chin, V. A. Diehl, and K. L. Norman, "Development of an Instrument Measuring User Satisfaction of the Human-computer Interface," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '88, 1988, pp. 213–218.

[466] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[467] N. Lettieri, A. Guarino, D. Malandrino, and R. Zaccagnino, "Platform Economy and Techno-Regulation - Experimenting with Reputation and Nudge," *Future Internet*, vol. 11, no. 7, p. 163, 2019.

[468] M. Lippi, G. Contissa, F. Lagioia, H.-W. Micklitz, P. Pałka, G. Sartor, and P. Torroni, "Consumer protection requires artificial intelligence," *Nature Machine Intelligence*, vol. 1, no. 4, pp. 168–169, 2019.

[469] F. Cozza, A. Guarino, F. Isernia, D. Malandrino, A. Rapuano, R. Schiavone, and R. Zaccagnino, "Hybrid and lightweight detection of third party tracking: Design, implementation, and evaluation," *Comput. Networks*, vol. 167, 2020.

[470] S. D'Ambrosio, S. D. Pasquale, G. Iannone, D. Malandrino, A. Negro, G. Patimo, V. Scarano, R. Spinelli, and R. Zaccagnino, "Privacy as a proxy for green web browsing: Methodology and experimentation," *Computer Networks*, vol. 126, pp. 81–99, 2017.

[471] M. Hildebrandt, *Law for Computer Scientists and Other Folk*. Oxford University Press, USA, 2020.

[472] A. Martín, H. D. Menéndez, and D. Camacho, "Mocdroid: multi-objective evolutionary classifier for android malware detection," *Soft Computing*, vol. 21, no. 24, pp. 7405–7415, 2017.

[473] R. Bronfman-Nadas, N. Zincir-Heywood, and J. T. Jacobs, "An artificial arms race: Could it improve mobile malware detectors?" in *2018 network traffic measurement and analysis conference (TMA)*. IEEE, 2018, pp. 1–8.

[474] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *arXiv preprint arXiv:1804.04637*, 2018.

[475] T. B. Lee, "The wannacry ransomware attack was temporarily halted. but itâs not over yet," 2017.

[476] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. van den Hengel, "Visual question answering: A survey of methods and datasets," *Computer Vision and Image Understanding*, vol. 163, pp. 21–40, 2017.

[477] H. Kaur, V. Mangat *et al.*, "A survey of sentiment analysis techniques," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2017, pp. 921–925.

[478] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[479] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 384–394.

[480] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[481] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[482] C. Wang, Z. Miao, Y. Lin, and J. Gao, "User and topic hybrid context embedding for finance-related text data mining," in *2019 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2019, pp. 751–760.

[483] Z. Yang and L. Li, "An online retrieval question answering system for featured snippets triggering," in *2019 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2019, pp. 49–55.

[484] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[485] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder for english," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 169–174.

[486] M. Chidambaram, Y. Yang, D. Cer, S. Yuan, Y.-H. Sung, B. Strope, and R. Kurzweil, "Learning cross-lingual sentence representations via a multi-task dual-encoder model," *arXiv preprint arXiv:1810.12836*, 2018.