To my parents

Contents

Introduction and methodology overview

The focus of this dissertation is the development and validation of a novel method for supervised text classification to be used effectively when small sized training sets are available. The proposed approach, which relies on a Weighted Word Pairs (WWP) structure, has been validated in two application fields: Query Expansion and Text Categorization.

By analyzing the state of the art for supervised text classification, it has been observed that existing methods show a drastic performance decrease when the number of training examples is reduced. This behaviour is essentialy due to the following reasons: the use, common to most existing systems, of the "Bag of Words" model where only the presence and occurrence of words in texts is considered, losing any information about the position; polysemy and ambiguity which are typical of natural language; the performance degradation affecting classification systems when the number of features is much greater than the available training samples.

Nevertheless, manual document classification is a boring, costly

and slow process: it has been observed that only 100 documents can be hand-labeled in 90 minutes and this number may be not sufficient for achieving good accuracy in real contexts with a standard trained classifier. On the other hand, in Query Expansion problems (in the domain of interactive web search engines), where the user is asked to provide a relevance feedback to refine the search process, the number of selected documents is much less than the total number of indexed documents. Hence, there's a great interest in alternative classification methods which, using more complex structures than a simple list of words, show higher efficiency when learning from a few training documents.

The proposed approach is based on a hierarchical structure, called Weighted Word Pairs (WWP), that can be learned automatically from a corpus of documents and relies on two fundamental entities: *aggregate roots* i.e. the words probabilistically more implied from all others; *aggregates* which are words having a greater probabilistic correlation with aggregate roots. WWP structure learning takes place through three main phases: the first phase is characterized by the use of probabilistic topic model and Latent Dirichlet Allocation to compute the probability distribution of words within documents: in particular, the output of LDA algorithm consists of two matrices that define the probabilistic relationship between words, topics and the documents. Under suitable assumptions, the probability of the occurrence of each word in the corpus, the conditional and joint probabilities between word pairs can be derived from these matrices. During the second phase, aggregate roots (whose number is selected by the user as an external parameter) are chosen as those words that maximize the condi-

tional probability product between a given word and all others, in line with the definition given above. Once aggregate roots have been chosen, each of them is associated with some aggregates and the coefficient of relationship between aggregate roots and aggregates is calculated thanks to the joint probability between word pairs (previously computed). The number of links between aggregate roots and aggregates depends on another external parameter (Max Pairs) which affects proper thresholds allowing to filter weakly correlated pairs. The third phase is aimed at searching the optimal WWP structure, which has to provide a synthetic representation for the information contained in all the documents (not only into a subset of them).

The effectiveness of the WWP structure was initially assessed in Query Expansion problems, in the context of interactive search engines. In this scenario, the user, after getting from the system a first ranking of documents in response to a specific query, is asked to select some relevant documents as a feedback, according to his information need. From those documents (relevance feedback), some key terms are extracted to expand the initial query and refine the search. In our case, a WWP structure is extracted from the relevance feedback and is appropriately translated into a query. The experimental phase for this application context was conducted with the use of TREC-8 standard dataset, which consists of approximately 520 thousand pre-classified documents. A performance comparison between the baseline (results obtained with no expanded query), WWP structure and a query expansion method based on the Kullback Leibler divergence was carried out. Typical information retrieval measurement were computed: preci-

sion at various levels, mean average precision, binary preference, R-precision. The evaluation of these measurements was performed using a standard evaluation tool used for TREC conferences. The results obtained are very encouraging.

A further application field for validating WWP structure is documents categorization. In this case, a WWP structure combined with a standard Information Retrieval module is used to implement a document-ranking text classifier. Such a classifier is able to make a soft decision: it draws up a ranking of documents that requires the choice of an appropriate threshold (Categorization Status Value) in order to obtain a binary classification. In our case, this threshold was chosen by evaluating performance on a *validation set* in terms of micro-precision, micro-recall and micro-F1. The dataset Reuters-21578, consisting of about 21 thousand newspaper articles, has been used; in particular, evaluation was performed on the ModApte split (10 categories), which includes only manually classified documents. The experiment was carried out by selecting randomly the 1% of the training set available for each category and this selection was made 100 times so that the results were not biased by the specific subset. The performance, evaluated by calculating the F1 measure (harmonic mean of precision and recall), was compared with the Support Vector Machines, in the literature referred as the state of the art in the classification of such a dataset. The results show that when the training set is reduced to 1%, the performance of the classifier based on WWP are on average higher than those of SVM.

This dissertation is structured as follows: in Chapter 1 the state of art on text classification and retrieval methods is dis-

cussed, together with a general modular framework for query expansion problems; in Chapter 2 the Weighted Word Pairs approach is presented in detail; in Chapter 3 performance evaluation of the proposed method is discussed in considered application fields; finally, in Chapter 4, conclusion and future works are drawn out.

"A goal is a dream with a finish line."

Duke Ellington

Chapter 1

Related Work

1.1 Text Classification problems

The problem of text classification has been widely studied in machine learning, database, data mining and information retrieval communities [2]. There are several application fields where text classification methods are usually employed: *News Filtering and Organization*, where large volumes of news articles are created everyday and need to be correctly categorized; *Document Organization and Retrieval*, which regards a broader range of cases such as categorization of digital libraries, web collections, scientific literature, social feeds; *Opinion Mining*, which is a hot topic nowadays because it regards classification of customer reviews and opinion to extract useful information for marketing purposes; *Email Classification and Spam Filtering*, where the aim is to determine junk email automatically. Several techniques have been proposed for

text classification tasks. Some of them exist also for other data domains such as quantitative data; in fact, text can be modeled as quantitative data with frequencies on the word attributes, although such attributes are tipically high dimensional with low frequencies on most of the words. Most used methods for text classification are: *Decision trees*, *Pattern-based Classifiers*, *SVM Classifiers*, *Neural Network Classifiers*, *Bayesian Classifiers* and other classifiers which can be adapted to the case of text data (nearest neighbor classifiers, genetic algorithm-based classifiers). An overview of such methods will be provided further in this chapter together with the *feature selection* problem, which aims to determine the features which are most relevant for the classification process.
Following the definition introduced in [70], a supervised *Text Classifier* may be formalized as the task of approximating the unknown target function $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ (namely the expert) by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ called the *classifier*, where $\mathcal{C} = \{\mathbf{c}_1, ..., \mathbf{c}_{|\mathcal{C}|}\}$ is a predefined set of *categories* and $\mathcal{D}$ is a set of *documents*.

If $\Phi(\mathbf{d}_m, \mathbf{c}_i) = T$, then $\mathbf{d}_m$ is called a positive example (or a member) of $\mathbf{c}_i$, while if $\Phi(\mathbf{d}_m, \mathbf{c}_i) = F$ it is called a negative example of $\mathbf{c}_i$.

Categories are just symbolic labels: no additional knowledge (of a procedural or declarative nature) regarding their meaning is usually available, and it is often the case that no metadata (such as e.g. publication date, document type, publication source) is available either. In these cases, the classification must be accomplished only through knowledge extracted from the documents themselves,

namely *endogenous knowledge*.

In practice, an initial corpus $\Omega = \{\mathbf{d}_1, \ldots, \mathbf{d}_{|\Omega|}\} \subset \mathcal{D}$ of documents pre-classified under $\mathcal{C} = \{\mathbf{c}_1, ..., \mathbf{c}_{|\mathcal{C}|}\}$ is considered. The values of the total function Φ are known for every pair $(\mathbf{d}_m, \mathbf{c}_i) \in \Omega \times \mathcal{C}$.

The initial corpus has to be split into two sets, not necessarily of equal size:

1. the *training* set: $\Omega_r = \{\mathbf{d}_1, \ldots, \mathbf{d}_{|\Omega_r|}\}$. The classifier Φ for the categories is inductively built by observing the characteristics of these documents;

2. the *test* set: $\Omega_e = \{\mathbf{d}_{|\Omega_r|+1}, \ldots, \mathbf{d}_{|\Omega|}\}$, used for testing the effectiveness of the classifiers.

In most problems, it can be simpler to consider the case of *single-label* classification, also called *binary*, where, given a category $\mathbf{c}_i$, each $\mathbf{d}_m \in \mathcal{D}$ has to be assigned either to $\mathbf{c}_i$ or to its complement $\overline{\mathbf{c}}_i$.

In fact, it has been demonstrated that, through transformation methods, it is always possible to transform the multi-label classification problem either into one or more single-label classification or regression problems [70, 75].

Therefore, the classification problem for $\mathcal{C} = \{\mathbf{c}_1, ..., \mathbf{c}_{|\mathcal{C}|}\}$ can be solved dealing with $|\mathcal{C}|$ independent classification problems for the documents in $\mathcal{D}$ under a given category $\mathbf{c}_i$, and so we have $\hat{\phi}_i$, for $i = 1, \ldots, |\mathcal{C}|$, classifiers. As a consequence, the whole problem in this case is to approximate the set of function $\Phi = \{\phi_1, \ldots, \phi_{|\mathcal{C}|}\}$ with the set of $|\mathcal{C}|$ classifiers $\hat{\Phi} = \{\hat{\phi}_1, \ldots, \hat{\phi}_{|\mathcal{C}|}\}$.

Since text cannot be directly interpreted by a classifier, an indexing procedure, that maps a text $\mathbf{d}_m$ into a compact representation of its content, must be uniformly applied to the training and test documents. Each document can be represented, following the *Vector Space Model* [18], as a vector of term *weights*

$$\mathbf{d}_m = \{w_{1m}, \ldots, w_{|\mathcal{T}|m}\},$$

where $\mathcal{T}$ is the set of *terms* (also called *features*) that occur at least once in at least one document of Ω_r, and $0 \leq w_{nm} \leq 1$ represents how much term t_n contributes to semantics of document $\mathbf{d}_{mm}$.

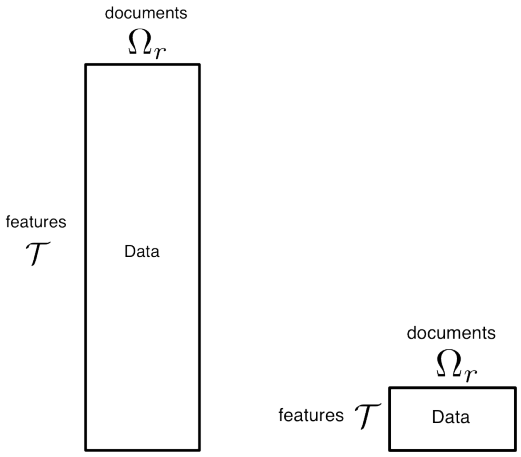
Identifying terms with words, we fall into the *bags of words* assumption where $t_n = v_n$, with v_n being a word of the vocabulary. The *bags of words* assumption claims that each w_{nm} indicates the presence (or absence) of a word, so that the information on the position of that word within the document is completely lost [18].

To determine the weight w_{nm} of term t_n in a document $\mathbf{d}_m$, the standard tf-idf (*term frequency-inverse document frequency*) function can be used [69], defined as:

$$\text{tf-idf}(t_n, \mathbf{d}_m) = N(t_n, \mathbf{d}_m) \cdot \log \frac{|\Omega_r|}{N_{\Omega_r(t_n)}} \tag{1.1}$$

where $N(t_n, \mathbf{d}_m)$ denotes the number of times t_n occurs in $\mathbf{d}_m$, and $N_{\Omega_r(t_n)}$ denotes the document frequency of term t_n, i.e. the number of documents in Ω_r in which t_n occurs.

In order for the weights to fall in the $[0, 1]$ interval and for the documents to be represented by vectors of equal length, the weights resulting from tf-idf are usually normalized by cosine normalization, given by:



(a) (b)

Figure 1.1 Features-documents matrix. 1.1(a) In this case the number of features is much higher than the number of examples ($|\mathcal{T}| \gg |\Omega_r|$). 1.1(b). In this case $|\mathcal{T}| \ll |\Omega_r|$.

$$w_{nm} = \frac{\text{tf-idf}(t_n, \mathbf{d}_m)}{\sqrt{\sum_{n=1}^{|\mathcal{T}|}(\text{tf-idf}(t_n, \mathbf{d}_m))^2}} \tag{1.2}$$

Word stopping (i.e. topic-neutral words such as articles, prepositions, conjunctions, etc.) and stemming procedures [1] (i.e. grouping words that share the same morphological root) are often performed during the indexing procedure so that a matrix $|\mathcal{T}| \times |\Omega_r|$ of real values is obtained instead of the training set Ω_r. The same procedure has to be applied to the test set Ω_e.

Usually, machine learning algorithms are susceptible to the problem named the *curse of dimensionality*, which refers to the degradation in the performance of a given learning algorithm as

[1] Although stemming has sometimes been reported to hurt effectiveness, the recent tendency is to adopt it as it reduces both the dimensionality of the feature space and the stochastic dependence between terms.

the number of features increases. In this case, the *computational cost* of the learning procedure and *overfitting* of the classifier are very common problems [8].

Moreover, from a statistical point of view, in the case of supervised learning, it is desirable that the number of labeled examples in the training set should significantly exceed the number of features used to describe the dataset itself.

In the case of text documents the number of features is usually high and particularly it is usually higher than the number of documents. In Fig. 1.1(a) we show the case of a training set composed of 100 documents and about 20000 features; note that $|\mathcal{T}| \gg |\Omega_r|$ while it is desirable to have the opposite condition, that is $|\mathcal{T}| \ll |\Omega_r|$, as represented in Fig. 1.1(b).

To deal with these issues, dimension *reduction techniques* are applied as a data pre-processing step or as part of the data analysis to simplify the whole data set (*global* methods) or each document (*local* methods) of the data set. As a result we can identify a suitable low-dimensional representation for the original high-dimensional data set, see Fig. 1.1(b).

In literature, we distinguish between methods that *select* a subset of the existing features or that *transform* them into a new reduced set of features. Both classes of methods can rely on a supervised or unsupervised learning procedure [8, 70, 18, 39]:

1. *feature selection*: $\mathcal{T}_s$ is a subset of $\mathcal{T}$. Examples of this are methods that consider the selection of only the terms that occur in the highest number of documents, or the selection of terms depending on the observation of information-theoretic

functions, among which we find the *DIA association factor*, *chi-square*, *NGL coefficient*, *information gain*, *mutual information*, *odds ratio*, *relevancy score*, *GSS coefficient* and others.

2. *feature transformation*: the terms in $\mathcal{T}_p$ are not of the same type as the terms in $\mathcal{T}$ (e.g. if the terms in $\mathcal{T}$ are words, the terms in $\mathcal{T}_p$ may not be words at all), but are obtained by combinations or transformations of the original ones. For example, there are methods that extract from the original a set of "synthetic" terms that maximize classification effectiveness; these are based on *term clustering*, *latent semantic analysis*, *latent dirichlet allocation*, *principal component analysis* and others. After a transformation we could need to reduce the number of the new features through a selection method thus obtaining a new set $\mathcal{T}_{sp}$ that is a subset of $\mathcal{T}_p$.

In the following section, some common feature selection methods are discussed in detail.

1.1.1 Feature selection

As introduced before, feature selection is very important in text classification due to the high dimensionality of text features and the existence of noise (irrelevant features). Typically, we can represent text in two ways: as a *bag of words*, where a document is represented as a set of words, each having an occurrence frequency, without minding the sequence; as *strings*, where each document is a sequence of words. However, most text classification methods

use the bag of words representation cause it's simpler. A common procedure used in both supervised and unsupervised application regards stop-words removal and stemming. In the first case, common words, which are not specific or discriminatory to different classes, are not indexed. In the second case different forms (singular, plural, different tenses) of the same word are consolidated into a single word.
One of the most common methods to quantify the discrimination level of a feature is the use of the *gini-index* measure. The *gini-index* $G(w)$ for the word w is defined as:

$$G(w) = \sum_{i=1}^{k} p_i(w)^2$$

where $p_i(w)$ is the conditional probability that a document belongs to class i, given that it contains the word w. Higher value of the *gini-index* indicate a great discriminative power of the word w (when all documents containing w belong to a class, we have the maximum value $G(w) = 1$).

Another common measure used for text feature selection is *information gain* or *entropy*. If $F(w)$ is the fraction of documents containing the word w, the information gain measure $I(w)$ for a given word w is defined as follows:

$$I(w) = -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \sum_{i=1}^{k} p_i(w) \cdot \log(p_i(w)) + \\ + (1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w))$$

where P_i is the global probability of class i, and $p_i(w)$ is the probability of class i, given that the document contains the word

w. Also in this case, a greater value of the information gain means a greater discriminative power.

The *mutual information measure* is another important measure for feature selection that derives from information theory. It is related to the level of co-occurrence between the class i and the word w. Specifically, we have:

$$M_i(w) = \log\left(\frac{p_i(w)}{P_i}\right)$$

Since $M_i(w)$ is specific to a particular class i, the overall mutual information as a function of the mutual information of word w with different classes has to be computed. This can be accomplished with the use of the average and maximum values of $M_i(w)$ over different classes:

$$M_{avg}(w) = \sum_{i=1}^{k} P_i \cdot M_i(w)$$

$$M_{max}(w) = \max M_i(w)$$

Either of these measures can be used to determine the relevance of the word w. A different way to compute the lack of independence between the word w and a particular class i is the χ^2 *statistic*. It is defined as follows:

$$\chi^2(w) = \frac{n \cdot F(w)^2 \cdot (p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i))}$$

Also for the χ^2 statistic a global value can be computed with the use of average and maximum value.

1.1.2 Feature Transformation

The aim of the *feature transformation* process is to create a new and smaller set of features as a function of the original set of

features. A typical method to accomplish this dimensionality reduction is Latent Semantic Indexing (LSI) and its probabilistic variant PLSA. The LSI method is able to transform a text space of few hundred thousand word features to a new axis system (composed of a few hundred features) which are a linear combination of the original word features. The axis system retaining more information about the variations in the underlying attribute values is determined through *Principal Component Analysis* techniques. Being an unsupervised technique, features found by LSI could not be the directions along which the class distribution of documents can be best separated. Better results for classification accuracy have been observed using boosting techniques in conjunction with the conceptual features obtained through pLSA and LDA (which is a Bayesian version of pLSA). A number of different methods have been proposed to adapt LSI to supervised classification. One common approach is to perform local LSI on the subsets of data representing the individual classes, and identify the discriminative eigenvectors from the different reductions with the use of an iterative approach [74]. This method is known as SLSI (Supervised Latent Semantic Indexing), and the advantages of the method seem to be limited; in fact the experiments in [74] show poor improvements over a standard SVM classifier, which did not use a dimensionality reduction process. A combination of class-specific LSI and global analysis is used in [76], where class-specific LSI representations are created. Test documents are compared against each LSI representation in order to create the most discriminative reduced space. Note that the different local LSI representations use a different subspace, so it is difficult to compare the similarities

of the different documents across the different subspaces. Furthermore, both the methods in [74] [76] tend to be computationally expensive. The "sprinkling" method is proposed in [17], in which artificial terms are added to the documents, which correspond to the class labels. In other words, a term corresponding to the class label is created and added to the document. LSI is then performed on the document collection with these added terms. The sprinkled terms can then be removed from the representation, once the eigenvectors have been determined. The sprinkled terms help in making the LSI more sensitive to the class distribution during the reduction process. In [17] an adaptive sprinkling process is also proposed, where all classes are not necessarily treated equally, but the relationships between the classes are used in order to regulate the sprinkling process. *Text clustering* is often used for feature transformation [57][72] . Clusters are created from a text collection thanks to supervision from the class distribution. Words that frequently occur in the supervised cluster can be used to create the new set of dimensions and classification can be performed according to this new feature representation. This approach retains interpretability with respect to the original words of the document collection but the optimum directions of separation may not be represented in the form of clusters of words and the underlying axes are not necessarily orthonormal to one another. Another common method is *Fisher's linear discriminant* [38]. This method is aimed to determine the directions in the data along which the points are as well separated as possible. The power of such a dimensionality reduction approach has been illustrated in [16], where it is shown that a simple decision tree classifier has better perfor-

mance on this transformed data when compared to more sophisticated classifiers. The *Topical Difference Factor Analysis* method also attempts to determine projection directions that maximize the topical differences between different classes and, when used with a k-nearest neighbor classifier, shows a great improvement of accuracy compared to the original set of features. A generalized dimensionality reduction method has been proposed in [45] [44] as an *unsupervised* method; it preserves the underlying structure assuming that data has been clustered in a pre-processing phase.

1.1.3 Dimension reduction methods

The *clustering* process is aimed at placing documents into groups relying on similarity information about them (if each document can be assigned to different clusters, we have *soft* clustering) [29]. A low dimensional representation for documents is then obtained since each cluster can be viewed as a dimension which is function of all original features. *Topic modeling* integrates soft clustering with dimension reduction: each document is assigned to a set of latent topics that correspond to both document clusters and compact representations identified from a corpus. The degree of membership between a document and the cluster is assessed through a weight which represent also a coordinate of the document in the reduced dimension space. Given a corpus of M documents, there will be W distinct terms of vocabulary and a term-document matrix X of size $W \times M$ can be defined: it encodes the occurrences of each term in each document. A multinomial distribution is commonly used for text modeling since it captures the relative

frequency of terms in a document with l_1-norm standardization. Since Dirichlet distribution is the conjugate distribution to multinomial, it is often used as a prior for multinomial models. In the following sections, *Latent Semantic Indexing* and *Latent Dirichlet Allocation* methods are briefly introduced as the most used methods for dimensionality reduction and topic modeling.

Latent Semantic Indexing

LSI projects both documents and terms into a low dimensional space which represents the semantic concepts in the document. This projection enables search engine to find documents containing the same concepts but different terms, overcoming issues of *synonym* and *polysemy*. LSI relies on singular value decomposition (SVD) of term-document matrix X : a low rank approximation of X, which has the effect of propagating co-occurring terms in the document corpus, is computed so that the dimensions of the approximation are interpreted as semantic concepts. The projections into latent semantic space can be used to perform several tasks more efficiently; in information retrieval, for example, a query can be viewed as a short document and projected in the latent semantic space: its similarity with the document is then measured in that space, mitigating problems of synonym and polysemy. Since real documents tend to be *bursty*, an uncommon term is likely to occur multiple times in a document if it occurs at all[19]. The use of term frequency would boost the contribution of such a term. There are two ways to address this problem. The first is to use a binary representation which takes into account if a term occurs in a document, ignoring its frequency. The second is to

use a global term-weight methods where term frequency is scaled with inverse document frequency (IDF)[18]. A language pyramid model [78] can also be used to provide a matrix representation for documents where not only term occurrence is taken into account but also spatial information such as term proximity, ordering, long distance dependence and so on. In order to handle changes in the corpus, which are frequent in real world applications, two techniques are often employed so that SVD does not need to be fully recomputed. The *Fold-in* method aims to compute the projection of the new documents and terms into an existing latent semantic space: it is very efficient but over time the outdated model becomes increasingly less useful. In [79] an interesting approach to update a LSI model is proposed and it is based on performing LSI on $[\hat{X}X']$ instead of XX', where X' is the term-document matrix for the new documents. In the same work, the authors show that this approximation don't introduce unacceptable errors. A good study about the optimal dimension of the latent semantic space can be found in [36], where the author shows how LSI can deal with the problem of synonymy and provides an upper bound for the dimension of latent semantic space, which allows to represent the corpus correctly.

Probabilistic Latent Semantic Indexing

PLSI [43] extends LSI in a probabilistic context using a probabilistic generative process to generate a word w in the document d of a corpus. An unobservable topic variable z is associated with each observation (v, d), where v is a term sample for token w. The joint probability distribution $p(v, d)$ can be expressed as

$p(v,d) = p(d)p(v|d)$, where

$$p(v|d) = \sum_{i=1}^{K} p(v|z=i)p(z=i|d)$$

The generative process assumes that: a document d is sampled from multinomial distribution $p(d)$; a topic i is sampled from the topic distribution $\theta_{di} = p(z=i|d)$; a term v for token w is sampled based on $\phi_{iv} = p(w=v|z=i)$. Typically, a different formulation (called *simmetric formulation*) is used to express the joint probability $p(v,d)$, which models documents and terms in a simmetric manner.

This is obtained by writing $p(z=i|d) = p(z=i)p(d|z=i)$ so that

$$p(v|d) = \sum_{i=1}^{K} p(z=i)p(v|z=i)p(d|z=i)$$

Note that $p(d|z=i)$ and $p(w|z=i)$ represent the projection of documents and terms into the latent semantic spaces (connection to LSI). Unknown parameters (probability distributions) are estimated by maximizing log-likelihood of observed data or minimizing Kullback-Liebler divergence [80] between the measured distribution $\hat{p}(v|d)$ and the model distribution $p(w|d)$. Since this is non-convex, expectation-maximization [30] is used to seek a locally optimal solution. Updating is usually obtained through *fold-in* method as in LSI: an EM algorithm is used to obtain $p(z|d)$ [42], while $p(w|z)$ and $p(z)$ are not updated.

Latent Dirichlet Allocation

In order to reduce the number of parameters to be learned and to provide a well defined probability for new documents, *Latent*

Dirichlet Allocation [7] has been proposed as an alternative to PLSI for text analysis: it includes a process for generating the topics in each document. According to LDA model (Figure 1.2), the distribution of terms for each topic i is represented as a multinomial distribution Φ_i drawn from a symmetric Dirichlet distribution with parameter β:

$$p(\Phi_i|\beta) = \frac{\Gamma(W\beta)}{[\Gamma\beta]^W} \prod_{v=1}^{W} \phi_{iv}^{\beta-1}$$

The topic distribution for document d is also represented as a multinomial distribution Θ_d drawn from a Dirichlet distribution with parameters α:

$$p(\Theta_d|\alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} \theta_{di}^{\alpha_i-1}$$

In this way, the topic z_{dn} for each token index n can be chosen from the document topic distribution as

$$p(z_{dn} = i|\Theta_d) = \theta_{di}$$

and each token w is chosen from the multinomial distribution associated with the selected topic

$$p(w_{dn} = v|z_{dn} = i, \Phi_i) = \phi_{iv}$$

LDA aims to find patterns of term co-occurrence in order to identify coherent topics. Note that if we use LDA to learn a topic i and we have that $p(w = v|z = i)$ is high for a certain term v, then any document d that contains term v has a high probability for topic i. We can say that all terms that co-occur with term v are more likely to have been generated by topic i.

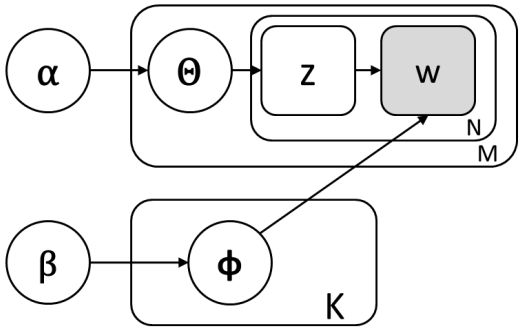


Figure 1.2 A diagram of LDA graphical model

In [60] influences of symmetry or asymmetry of Dirichlet priors on the mechanism are discussed. Authors show that a symmetric prior provides smoothing for topic-specific term distributions so that unseen terms don't have zero probability. Otherwise, an asymmetric prior for the document-specific topic distribution makes LDA more robust to stopwords and less sensitive to the selection of the number of topics resulting in more stable behaviour. Standard LDA tends to learn broad topics. If a topic has several aspects, each of them will co-occur frequently with the main concept and LDA will come out with a topic including the concept and all aspects. Other concepts are progressively added to the same topic if they share the same aspects and the topics become diffuse. When sharper topics are needed, a hierarchical topic model could be more appropriate.

In order to train an LDA model, it is necessary to find the optimal set of parameters to maximize the probability of generating the training documents. Such a probability is called *empirical likelihood* and it is hard to optimize directly since the topic assignments z_{dn} cannot be observed. Therefore, two approximations for LDA

are commonly used where as exact inference is intractable: *Collapsed Gibbs* and *Variational Approximation*. In Gibbs sampling, random values are first assigned to each variable, which is then sampled in turn conditioned on the value of the other variables. The process explores several configurations according to the number of iterations and estimates underlying distributions. *Collapsed* Gibbs sampling is proposed in [41] with Θ and Φ marginalized. A topic is selected for a word if it is frequently used in the document or if is frequently assigned for the same term corpus-wide. After a burn in period, where a large number of samples is rejected, the procedure keep statistics of the number of times that each topic is selected for each word and, after an aggregating and normalizing phase, topic distributions for each document are estimated. An alternative to Gibbs sampling in training LDA models is represented by the variational approximation. Variational inference approximates the true posterior distribution of the latent variables by a fully factorized distribution (the variational model) where all the latent variables are independent of each other. The variational distribution can be viewed as a simplification of the original LDA graphical mode shown in Figure 1.2, where the edges between the nodes Θ and Z are ignored. A detailed discussion of LDA model can be found in [7].

1.1.4 Feature selection and Classification

It is well known that classification and feature selection processes are dependent upon one another, so it can be useful to investigate how the feature selection process interacts with classification algo-

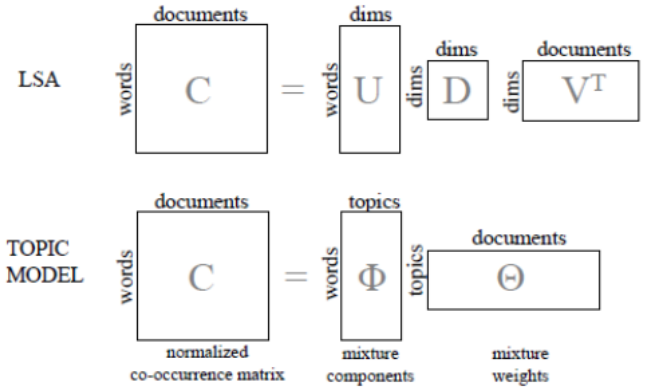


Figure 1.3 LSA and LDA representations.

rithms. Common issues in this context regard: the use of intermediate results form classification algorithms to create feature selection methods that can be used by other classification algorithms; the performance comparison of different feature selection methods used in conjunction with different classification algorithms. In [60] is shown that feature selection derived from linear classifiers provides very effective results. Moreover, the sophistication of the feature selection process itself is more important than the specific pairing between the feature selection process and the classifier.

In *Linear Classifiers* for example, the output of the linear predictor is defined as $p = \mathbf{A} \cdot \mathbf{X} + b$, where $\mathbf{X} = (x_1, \ldots, x_n)$ is the normalized document word frequency vector, $\mathbf{A} = (a_1, \ldots, a_n)$ is a vector of linear coefficients with the same dimensionality as the feature space and b is a scalar. If the coefficient a_i is close to zero, then the corresponding feature is assumed not to have a significant effect on the classification process. Otherwise, large values for a_j suggest that such a feature should be selected for classification. It has been shown that feature selection methods derived from linear

classifiers, perform well also when used with non linear classifiers.

1.1.5 Decision Tree Classifiers

In decision trees, a *predicate* or a condition on the attribute value is used to divide the data space hierarchically. As regards classification of text data, such predicates are typically condition on the presence or absence of one or more words in the documents. The division of the data space, performed recursively, terminates when the leaf nodes contain a certain minimum number of records or some condition on class purity; the majority class label is used for the task classification. The sequence of predicates is applied at the nodes in order to traverse a path of the tree from top to bottom and determine the relevant leaf node. Some of the nodes may be pruned in order to reduce overfitting, by holding out a part of the data not used to construct the tree. If the class distribution in the training data differs significantly from the class distribution in the data used for pruning, then it is assumed that the node overfits the training data and it has to be pruned. In the case of text data, the predicates for the decision tree are defined by considering the terms in the underlying text collection: a node could be partitioned into its children because of the presence or absence of a particular term in the document. There are different kind of possible splits: in *Single Attribute Splits*, the presence of a word, which provides the maximum discrimination between classes, at a particular node is used to perform the split (measures such as the gini-index or information gain are also used to perform the split); in *Similarity-based multi-attribute split*, similarity of documents to

frequent word clusters is typically used, where documents are further partintioned into groups and ordered by similarity value; in *Discriminant-based multi-attribute split*, the Fisher discriminant is often used to provide the directions in the data along which the classes are best separated. A detailed discussion of decision tree methods is found in [9, 1, 46, 64].

1.1.6 Rule based Classifiers

Rule based classifiers attempt to model the data space with a set of rules, where the left hand side is a condition on the underlying feature set and the right hand side is tthe class label. The rule set is extracted from the training data. A predicted class label is determined as a function of the class labels of the rules which are satisfied by the test instance. In most cases, the condition on the left side represents a set of terms which must be present in the document for the condition to be satisfied. Note that the *set intersection* of conditions on term presence is much more used than the union. In fact, in the case of union between conditions, each rule can be always split in two separate rules, each containing more information. While decision trees attempt to partion data space in a hierarchical fashion, rule based classifiers allow overlaps in the decision space. The idea is to create a rule set , such as all points in the decision space are covered by at least one rule. This can be achieved with the generation of a set of targeted rules which are related to the different classes and one default rule which can cover the remaining instances.

The two most common criteria to generate rules from training

data are *support* and *confidence*. Support quantifies the absolute number of instances in the training data which are relevant to the rule. Confidence quantifies the conditional probability that the right hand side of the rule is satisfied if the left hand side is satisfied. Since overlaps are allowed, it is possible that more than one rule is relevant to test the instance. In such a case, a rank-ordering of the rules is needed [58]. A common approach is to rank-order the rules by their confidence and pick the top-k rules as the most relevant. As regards text data, an interesting proposal for rule-based classification is in [3], where an iterative methodology is used for generating rules. Another important rule-based technique is RIPPER [25, 24], which treat documents as set-valued objects and generate rules based on the co-presence of the words in the documents. This method has been shown to be especially effective in scenarios where the number of training examples is relatively small.

1.1.7 Probabilistic and Naive Bayes Classifier

In probabilistic classifiers, an implicit mixture model for generation of the underlying documents is used. This mixture model assumes that each class is a component of the mixture, where a component is a generative model providing the probability of sampling a particular term for that component or class. The *Naive Bayes* classifier is the most commonly used generative classifier: the distribution of documents in each class is modeled using a probabilistic model where indipendence assumptions about the distributions of terms is made. Typically, models used for Naive

Bayes Classification compute the posterior probability of a class relying on the distribution of the words in the documents. The already discussed "bag of words" assumption is made so that the models ignore the actual position of the words in the document. The two mainly used model in Naive Bayes Classification differs for the assumption of taking or not taking word frequencies into account and also for the approach used for sampling the probability space. In the *Multivariate Bernoulli Model*, the presence or absence of words in a text document is used as a feature for document representation. Therefore, we don't use word frequencies to model the document, but the word features are assumed to be binary, we only have to indicate presence or absence of a word in the text. In the *Multinomial Model*, term frequencies are captured, representing the document as a bag of words. The document in each class can the be modeled as samples drawn from a multinomial word distribution. The conditional probability of a document given a class is the product of the probability of each observed word in the corresponding class. Once documents in each class have been modeled, the component class models together with the Bayes rule are used to compute the posterior probability of the class for a given document, and the class with the highest posterior posterior probability can be then assigned to the document. Note that methods which generalize the naive Bayes classifier by not using independence assumption don't work well because they have higher computational costs and are not able to estimate the parameters accurately and robustly in presence of limited data [66]. Although the indipendence assumption is a practical approximation, [27, 31] show that such an approach has theoretical

merit and naive classification work well in practice. Several papers [15, 49, 53, 56] show the use of Naive Bayes approach in a number of different application domain, also in cases where the importance of a document may decay with time [68]. A particular domain shown in [67] regards the filtering of junk mail. For this problem, we may have some additional knowledge to be incorporated in the process to help us determine if a particular message is junk or not. Some characteristics could be: a particular domain in sender address; the presence of emphasized punctuation following phrases such as "Free Money"; the recipient of the message was a particular user or mailing list. Bayesian Methods allow to incorporate such additional information by creating new features for each of these characteristics. Note that also hyperlink information can be incorporated into the classification process as shown in [14, 62]. In hierarchical classification problems, a Bayes classifier can be built at each node, providing the next branch to follow for the classification task. It has been observed that context-sensitive feature selection generally provides more useful classification. An information-theoretic approach [28] is used in work [53] for feature selection: it takes into account the dependencies between the attributes and features are progressively eliminated. An extensive comparison between the bernoulli and the multinomial models on different dataset has been performed in the work [59]. The multi-variate Bernoulli model sometimes performs better than the multinomial model when the size of the vocabulary is small. The multinomial model outperforms the multivariate Bernoulli model for large vocabulary sizes and has a better behaviour than the multi-variate Bernoulli when vocabulary size is chosen optimally

for both.

1.1.8 SVM Classifiers

Support Vector Machines are *Linear Classifiers* which attempt to determine linear separators between different classes. Typically, linear classifiers are strictly related to many feature transformation methods which use directions to transform the feature space and use other classifier on the transformed feature space. The SVM method attempts to determine the optimum direction of discrimination in the feature space by examining the appropriate combination of features, so it is quite robust when dealing with high dimensionality. Text data is well suited for SVM classification because of the sparse high-dimensional nature of text: features are highly correlated and organized in categories which can be linearly separated. Linear SVM is often used thanks to its simplicity and ease of interpretability. The first use of SVM in text classification was proposed in [49, 50] while a theoretical study is shown [51] and emphasizes why SVM classifier is expected to work well in different conditions. It has been shown that SVM approach provides better performance in *spam* classification when compared to other techniques such as boosting decision trees, the rule based RIPPER method and the Rocchio method [32]. It can also be combined with interactive user-feedback methods. Since the aim of these methods is to find the best separator, we have to deal with an optimization problem that can be reduced in most cases to a Quadratic Programming (QP) Problem. Newton's method for iterative minimization of a convex function is often used al-

though it can be slow for high dimensional domains (text data). Anyway, a large QP problem can be break into a set of smaller problems in order to find a solution in a more efficient manner [35]. The SVM approach has been used successfully in context of hierarchical organization of the classes [33] and in scenarios where a large amount of unlabeled data and a small amount of labeled data is available [71].

1.1.9 The Rocchio Framework

Distance-based measures can be used for classification purposes. This is the case of proximity-based classifiers, where measures such as the dot product or the cosine metric are used in order to assign a document to a class or to its complement [69]. More in general, in the domain of text classification, two main methods are often used: the first one aims to determine the k-nearest neighbors in the training data to the test document. The class label is selected by evaluating the majority class from the k neighbors, with k typically varying between 20 and 40 [23]; the second method relies on a pre-processing phase where clusters of training document belonging to the same class are created. After a representative meta-document is obtained from each group, the k-nearest neighbor approach is applied to the set of meta documents [54].
The most basic among methods which use grouping techniques for classification has been proposed by Rocchio in [65]. After a single representative meta-document is extracted from each class, the weight of a term t_k, for a given class, is the normalized frequency of the term t_k in the documents belonging to that class, minus the

normalized frequency of the term in documents which do not belong to that class. Being f_p^k the expected weight of the term t_k in a randomly chosen document belonging to the positive class, and f_n^k the corresponding for the negative class, for weighting parameters α_p and α_n, the weight f_{rocchio}^k is defined as follows:

$$f_{\text{rocchio}}^k = \alpha_p \cdot f_p^k - \alpha_n \cdot f_n^k$$

The weighting parameters α_p and α_n are chosen so that the positive class has a greater impact than the negative class. For the relevant class, a vector representation of the terms $(f_{\text{rocchio}}^1, \ldots, f_{\text{rocchio}}^n)$ is then obtained. Once the approach has been applied to each class obtaining $|C|$ meta-documents, the closest meta-document to the test document can be determined by using a vector-based dot product or other similarity metric. This class of methods, which create a profile for an entire class, is referred as the *Rocchio Framework*. This method is very simple and efficient but has a main drawback: if a single class occurs in multiple disjoint clusters (not well connected in the data), the centroid of these examples may not represent the class behaviour well. A detailed analysis of the Rocchio algorithm can be found in [49].

1.2 Text Retrieval problems

In the field of text retrieval the main problem is: "How can a system tell which documents are relevant to a query? Which results are more relevant than others?" To answer these questions, several Information Retrieval models have been proposed: set-theoretic (including boolean), algebraic and probabilistic models

[18][4]. Although each method has its own properties, there is a common denominator: the *bag of words* approach to document representation.

As explained in a previous section, the "bag of words" assumption claims that a document can be considered as a feature vector where each element indicates the presence (or absence) of a word, so that the information on the position of that word within the document is completely lost [18]. The elements of the vector can be weights (computed in several ways) so that a document can be viewed as a list of weighted features. The *term frequency-inverse document (tf-idf)* model is a commonly used weighting model: each term in a document collection is weighted by measuring how often it is found within a document (*term frequency*), offset by how often it occurs within the entire collection (*inverse document frequency*). Note that a query can be viewed as a document, so it can be represented as a vector of weighted words too. So the relevance of a document to a query can be measured as a distance between the corresponding vector representations in the features space. Unfortunately, queries performed by users may not be long enough [48][47] to avoid the inherent ambiguity of language (polysemy etc.). This makes text retrieval systems, that rely on a term-frequency based index, generally suffer from low precision, or low quality document retrieval.

To overcome this problem, scientists proposed methods to expand the original query with other topic-related terms. The idea of taking advantage of additional knowledge to retrieve relevant documents has been largely discussed in the literature, where manual, interactive and automatic techniques have been proposed

[37][18][4]. A better specialization of the query can be obtained with additional knowledge, that can be extracted from *exogenous* (e.g. ontology, WordNet, data mining) or *endogenous* knowledge (i.e. extracted only from the documents contained in the collection) [5][18].

1.2.1 The Relevance Feedback

In this dissertation, the focus is mainly on those query expansion techniques which make use of the *Relevance Feedback* (in the case of endogenous knowledge). In the literature we can distinguish between three types of procedures for relevance assignment: explicit feedback, implicit feedback, and pseudo feedback [4]. The feedback is usually obtained from assessors and indicates the relevance degree for a document retrieved in response to a query. If the assessors know that the provided feedback will be used as a relevance judgment then the feedback is called *explicit*. *Implicit* feedback is otherwise inferred from user behavior: it takes into account which documents they do and do not select for viewing, the duration of time spent viewing a document, or page browsing or scrolling actions. Pseudo relevance feedback (or *blind* feedback) assumes that the top "n" ranked documents obtained after performing the initial query are relevant: this approach is generally used in automatic systems. Since human labeling task is enormously boring and time consuming [52], most existing methods make use of pseudo relevance feedback. Nevertheless, fully automatic methods suffer from obvious errors when the initial query is intrinsically ambiguous. As a consequence, in recent years, some

hybrid techniques have been developed which take into account a minimal explicit human feedback [63][34] and use it to automatically identify other topic related documents: such methods achieve a mean average precision of about 30% [63].

However, whatever the technique that selects the set of documents representing the feedback, the expanded terms are usually computed by making use of well known approaches for term selection as Rocchio, Robertson, CHI-Square, Kullback-Lieber etc [13]. In this case the reformulated query consists in a simple (sometimes weighted) list of words. Although such term selection methods have proven their effectiveness in terms of accuracy and computational cost, several more complex alternative methods have been proposed, which consider the extraction of a structured set of words instead of simple list of them: a weighted set of clauses combined with suitable operators [12][26][55]. Since the aim of this dissertation is to validate a novel feature extraction approach in text retrieval problems, a general query expansion framework will be presented in detail through the next section.

1.2.2 A general query expansion framework

A general query expansion framework can be described as a modular system including one or several instances, properly combined, of:

- an Information Retrieval (IR) module;
- a Feedback (F) module;
- a Feature Extraction (FE) module;

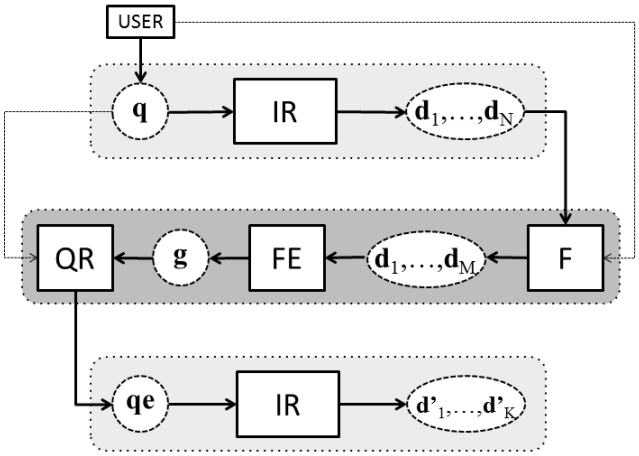


Figure 1.4 General framework for Query Expansion.

- a Query Reformulation (QR) module.

A common framework is represented in Figure 1.4 and can be explained as follows. The user initially performs a search task on the dataset $\mathcal{D}$ by inputting a query $\mathbf{q}$ to the IR system. A set of documents $\mathcal{RS} = (\mathbf{d}_1, \cdots, \mathbf{d}_N)$ is obtained as a result.

The module F identifies a small set of relevant documents $\mathcal{RF} = (\mathbf{d}_1, \cdots, \mathbf{d}_M)$ from the hit list of documents $\mathcal{RS}$ returned by the IR system. In case of explicit relevance feedback, we assume that module F requires user interaction. Given the set of relevant document $\mathcal{RF}$, the module FE extracts a set of features $\mathbf{g}$ that must be added to the initial query $\mathbf{q}$. The extracted features can be weighted words or more complex structures such as weighted word pairs. So the obtained set $\mathbf{g}$ must be adapted by the QR module to be handled by the IR system and then added to the initial query. The output of this module is a new query $\mathbf{qe}$ which includes both the initial query and the set of features

extracted from the $\mathcal{RF}$. The new query is then performed on the collection so obtaining a new result set $\mathcal{RS}' = (\mathbf{d}'_1, \cdots, \mathbf{d}'_K)$, obviously different from the one obtained before.

Considering the framework described above is possible to take into account any technique of feature extraction that makes use of relevance feedback and any IR systems suitable to handle the resulting expanded query **qe**. In this way it is possible to implement several techniques and make objective comparisons with the proposed one.

Following the theory behind these IR systems, queries and documents representation is based on the *Vector Space Model* [18], that considers vectors of weighted terms belonging to a vocabulary $\mathcal{T}$:

$$\mathbf{d} = \{w_1, \ldots, w_{|\mathcal{T}|}\}.$$

Each weight w_n is such that $0 \leq w_n \leq 1$ and represents how much the term t_n contributes to the semantics of the document $\mathbf{d}$ (in the same way for $\mathbf{q}$). Although each system has its own weighting function, tf-idf [40] for Lucene and statistical language modeling [61] for Indri, the weight is typically proportional to the term frequency and inversely proportional to the frequency and length of the documents containing the term.

Given a query, the IR system assigns the importance to each document of the collection by using the similarity function as defined in the following:

$$sim(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q} \cap \mathbf{d}} w_{t,\mathbf{q}} \cdot w_{t,\mathbf{d}}, \tag{1.3}$$

where $w_{t,\mathbf{q}}$ and $w_{t,\mathbf{d}}$ are the weights of term t in the query $\mathbf{q}$ and document $\mathbf{d}$.

In the following chapter a novel feature extraction approach, called *Weighted Word Pairs*, is presented and discussed in detail. Such a method, when employed in query expansion problems, makes use of explicit relevance feedback and achieves good performance even when few documents are selected as user feedback; it is based on a structured representation that can be automatically extracted from the documents of the minimal explicit feedback using a method of *term extraction* [22][21] based on the *Latent Dirichlet Allocation* model [7] implemented as the *Probabilistic Topic Model* [41].

The proposed approach has been validated using IR systems that allow to handle structured queries composed of weighted word pairs. For this reason, the following open source tools were considered: Apache Lucene [40] which supports structured query based on a weighted boolean model and Indri Lemur Toolkit [61] which supports an extended set of probabilistic structured query operators based on INQUERY.

"Anyone can make the simple complicated. Creativity is making the complicated simple."

CHARLES MINGUS

Chapter 2

The Weighted Word Pairs Approach

2.1 Introduction

The aim of the proposed method is to extract from a corpus of documents a compact representation, named *Weighted Word Pairs* (WWP), which contains the most discriminative word pairs to be used in text retrieval or classification tasks. The Feature Extraction module (FE) is represented in Fig. 2.1. The input of the system is the set[1] of documents:

$$\mathcal{RF} = \Omega_r = (\mathbf{d}_1, \cdots, \mathbf{d}_M)$$

[1]The relevance feedback $\mathcal{RF}$ can be interpreted as the training set Ω_r for the feature extraction module.

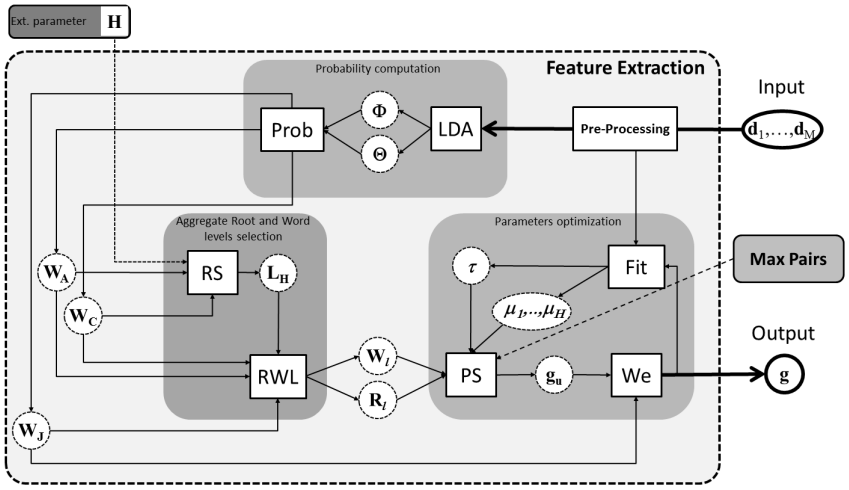


Figure 2.1 Proposed feature extraction method. A *Weighted Word Pairs* **g** structure is extracted from a corpus of training documents.

and the output is a vector of weighted word pairs:

$$\mathbf{g} = \{w'_1, \cdots, w'_{|\mathcal{T}_p|}\}$$

where $\mathcal{T}_p$ is the number of pairs and w'_n is the weight associated to each pair (feature) $t_n = (v_i, v_j)$. Note that a *feature transformation* process is involved: it turns word pairs, instead of single words, into basic features. Being $|\mathcal{T}|$ the basic feature set, a new space $|\mathcal{T}_p| \propto |\mathcal{T}|^2$ of features is obtained and need to be properly reduced to a subset $\mathcal{T}_{sp}$ such that $|\mathcal{T}_{sp}| \ll |\mathcal{T}_p|$. The *pre-processing* phase helps reduce the size of the basic feature set (vocabulary) by performing stopwords filtering and stemming. As further explained, the method used to select the most representative word pairs among all the $|\mathcal{T}_p|$ is based on the *Latent Dirichlet Allocation* [7] implemented as the *Probabilistic Topic Model* [41].

A WWP structure can be suitably represented as a *graph* $\mathbf{g}$ of terms (Fig. 2.2). Such a graph is made of several clusters, each containing a set of words v_s (*aggregates*) related to an *aggregate root* (r_i), a special word which represents the centroid of the cluster. How aggregate roots are selected will be clear further. The weight ρ_{is} can measure how a word is related to an aggregate root and can be expressed as a probability: $\rho_{is} = P(r_i|v_s)$. The resulting structure is a subgraph rooted on r_i. Moreover, *aggregate roots* can be linked together building a centroids subgraph. The weight ψ_{ij} can be considered as a degree of correlation between two aggregate roots and can also be expressed as a probability: $\psi_{ij} = P(r_i, r_j)$. Being each aggregate root a special word, it can be stated that $\mathbf{g}$ contains pairs of features lexically denoted as words.

Given the training set Ω_r of documents, the term extraction procedure is obtained first by computing all the relationships between words and aggregate roots (ρ_{is} and ψ_{ij}), and then selecting the right subset of pairs $\mathcal{T}_{sp}$ from all the possible ones $\mathcal{T}_p$. Before explaining in detail the learning procedure of a WWP graph, some aspects of this representation are clarified.

2.1.1 Graph and document representation in the space $\mathcal{T}_{sp}$

As introduced before, a WWP structure $\mathbf{g}$ can be viewed, following the *Vector Space Model* [18], as a vector of features t_n:

$$\mathbf{g} = \{b_1, \ldots, b_{|\mathcal{T}_{sp}|}\},$$

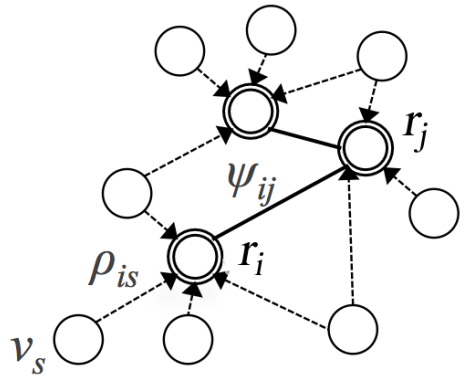


Figure 2.2 Graphical representation of a *Weighted Word Pairs* structure.

where $|\mathcal{T}_{sp}|$ represents the number of pairs and each feature $t_n = (v_i, v_j)$ can be a *word/aggregate root* or *aggregate root/aggregate root* pair. The weight b_n is named *boost* factor and is equal to ψ_{ij} for both *word/aggregate root* or *aggregate root/aggregate root* pairs.

Moreover, by following this approach, each document of a corpus can be represented in terms of pairs:

$$\mathbf{d}_m = (w_{1m}, \ldots, w_{|\mathcal{T}_{sp}|m}),$$

where w_{nm} is such that $0 \leq w_{nm} \leq 1$ and represents how much term $t_n = (v_i, v_j)$ contributes to a semantics of document $\mathbf{d}_m$. The weight is calculated thanks to the tf-idf model applied to the pairs represented through t_n:

$$w_{nm} = \frac{\text{tf-idf}(t_n, \mathbf{d}_m)}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|}(\text{tf-idf}(t_n, \mathbf{d}_m))^2}} \tag{2.1}$$

2.1.2 WWP-based classifier definition in the space $\mathcal{T}_{sp}$

If we learn a graph $\mathbf{g}_i$ from documents that are labeled as $\mathbf{c}_i$, then $\mathbf{g}_i$ is a representation of such labeled set of documents and can be considered as the expert $\hat{\phi}_i$ for the category $\mathbf{c}_i$:

$$\mathbf{g}_i = \hat{\phi}_i = \{b_{1i}, \ldots, b_{|\mathcal{T}_sp|i}\}.$$

Using the expert we can perform a classification task by using a linear method that measures the similarity between the expert $\hat{\phi}_i$ and each document $\mathbf{d}_m$ represented in the space $\mathcal{T}_{sp}$.

A text-ranking classifier, also called *soft decision* based classifier, is then obtained: for the category $\mathbf{c}_i \in \mathcal{C}$ we define a function (the cosine similarity) which, given a document $\mathbf{d}_m$, returns a *categorization status value* $CSV_i(\mathbf{d}_m) \in [0, 1]$:

$$CSV_i(\mathbf{d}_m) = \frac{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{ni} \cdot w_{nm}}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{ni}^2} \cdot \sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} w_{nm}^2}} \tag{2.2}$$

Such a number represents the evidence for $\mathbf{d}_m \in \mathbf{c}_i$; it is a measure of vector closeness in a $|\mathcal{T}_{sp}|$-dimensional space.

2.2 Building a WWP graph

A WWP graph $\mathbf{g}$ is learnt from a corpus of documents as a result of two important phases: the *Relations Learning* stage, where graph relation weights are learnt by computing probabilities between word pairs (see Fig. 2.1); the *Structure Learning* stage, where the shape of an initial WWP graph, composed by all possible aggregate root and word levels, is optimized by performing

an iterative procedure. The algorithm, given the number of aggregate roots H and the desired max number of pairs as constraints, chooses the best parameter settings τ and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_H)$ defined as follows:

1. τ: the threshold that establishes the number of *aggregate root/aggregate root* pairs of the graph. A relationship between the aggregate root v_i and aggregate root r_j is relevant if $\psi_{ij} \geq \tau$.

2. μ_i: the threshold that establishes, for each aggregate root i, the number of *aggregate root/word* pairs of the graph. A relationship between the word v_s and the aggregate root r_i is relevant if $\rho_{is} \geq \mu_i$.

2.2.1 Relations Learning

Since aggregate roots and aggregates are lexically represented as words of the vocabulary, we can write $\rho_{is} = P(r_i|v_s) = P(v_i|v_s)$, and $\psi_{ij} = P(r_i, r_j) = P(v_i, v_j)$.

Considering that $P(v_i, v_j) = P(v_i|v_j)P(v_j)$, all the relations between words result from the computation of the joint or the conditional probability $\forall i, j \in \{1, \cdots, |\mathcal{T}|\}$ and $P(v_j)$ $\forall j$.

An exact calculation of $P(v_j)$ and an approximation of the joint, or conditional, probability can be obtained through a smoothed version of the generative model introduced in [7] called Latent Dirichlet Allocation (LDA), which makes use of Gibbs sampling [41].

The original theory introduced in [41] mainly proposes a semantic representation in which documents are represented in terms

of a set of probabilistic topics z.

Formally, we consider a word u_m of the document $\mathbf{d}_m$ as a random variable on the vocabulary $\mathcal{T}$ and z as a random variable representing a topic between $\{1, \cdots, K\}$. A document $\mathbf{d}_m$ results from generating each of its words. To obtain a word, the model considers three parameters assigned: α, η and the number of topics K. Given these parameters, the model chooses θ_m through $P(\theta|\alpha) \sim \text{Dirichlet}(\alpha)$, the topic k through $P(z|\theta_m) \sim \text{Multinomial}(\theta_m)$ and $\beta_k \sim \text{Dirichlet}(\eta)$. Finally, the distribution of each word given a topic is $P(u_m|z, \beta_z) \sim \text{Multinomial}(\beta_z)$.

The output obtained by performing Gibbs sampling on a set of documents Ω_r consists of two matrixes:

1. the *words-topics* matrix that contains $|\mathcal{T}| \times K$ elements representing the probability that a word v_i of the vocabulary is assigned to topic k: $P(u = v_i|z = k, \beta_k)$;

2. the *topics-documents* matrix that contains $K \times |\Omega_r|$ elements representing the probability that a topic k is assigned to some word token within a document $\mathbf{d}_m$: $P(z = k|\theta_m)$.

The probability distribution of a word within a document $\mathbf{d}_m$ of the corpus can be then obtained as:

$$P(u_m) = \sum_{k=1}^{K} P(u_m|z = k, \beta_k)P(z = k|\theta_m). \tag{2.3}$$

In the same way, the joint probability between two words u_m and y_m of a document $\mathbf{d}_m$ of the corpus can be obtained by assuming that each pair of words is represented in terms of a set of

topics z and then:

$$P(u_m, y_m) = \sum_{k=1}^{K} P(u_m, y_m | z = k, \beta_k) P(z = k | \theta_m) \tag{2.4}$$

Note that the exact calculation of Eq. 2.4 depends on the exact calculation of $P(u_m, y_m | z = k, \beta_k)$ that cannot be directly obtained through LDA. If we assume that words in a document are conditionally independent given a topic, an approximation for Eq. 2.4 can be written as:

$$P(u_m, y_m) \simeq \sum_{k=1}^{K} P(u_m | z = k, \beta_k) P(y_m | z = k, \beta_k) P(z = k | \theta_m). \tag{2.5}$$

Moreover, Eq. 2.3 gives the probability distribution of a word u_m within a document $\mathbf{d}_m$ of the corpus. To obtain the probability distribution of a word u independently of the document we need to sum over the entire corpus:

$$P(u) = \sum_{m=1}^{M} P(u_m) \delta_m \tag{2.6}$$

where δ_m is the prior probability for each document (note that $\sum_{m=1}^{|\Omega r|} \delta_m = 1$).

In the same way, if we consider the joint probability distribution of two words u and y, we obtain:

$$P(u, y) = \sum_{m=1}^{M} P(u_m, y_v) \delta_m \tag{2.7}$$

Concluding, once we have $P(u)$ and $P(u, y)$ we can compute $P(v_i) = P(u = v_i)$ and $P(v_i, v_j) = P(u = v_i, y = v_j)$, $\forall i, j \in$

$\{1, \cdots, |\mathcal{T}|\}$ and so the relations learning can be totally accomplished.

2.2.2 Structure Learning

Once each ψ_{ij} and ρ_{is} is known $\forall i, j, s$, aggregate root and word levels have to be identified in order to build a starting WWP structure to be further optimized. The first step is to select from the words of the indexed corpus a set of aggregate roots $\mathbf{r} = (r_1, \ldots, r_H)$, which will be the nodes of the centroids subgraph. Aggregate roots are meant to be the words whose occurrence is most implied by the occurrence of other words of the corpus, so they can be chosen as follows:

$$r_i = \arg\max_{v_i} \prod_{j \neq i} P(v_i|v_j)$$

Since relationships' strenghts between aggregate roots can be directly obtained from ψ_{ij}, the centroids subgraph can be easily determined. Note that not all possible relationships between aggregate roots are relevant: the threshold τ can be used as a free parameter for optimization purposes. As discussed before, several words (aggregates) can be related to each aggregate root, obtaining H aggregates' subgraphs. The threshold set $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_H)$ can be used to select the number of relevant pairs for each aggregates' subgraph. Note that a relationship between the word v_s and the aggregate root r_i is relevant if $\rho_{is} \geq \mu_i$, but the value ρ_{is} cannot be directly used to express relationships' strenghts between aggregate roots and words. In fact, being ρ_{is} a conditional probability, it is always bigger than ψ_{is} which is a joint probability.

Therefore, once pairs for the aggregates' subgraph are selected using ρ_{is}, relationships' strenghts are represented on the WWP structure through ψ_{is}.
Given H and the maximum number of pairs as constraints (i.e. fixed by the user), several WWP structure $\mathbf{g}_t$ can be obtained by varying the parameters $\Lambda_t = (\tau, \boldsymbol{\mu})_t$.

As shown in Fig.2.1, an optimization phase is carried out in order to search the set of parameters Λ_t which produces the best WWP graph. This process relies on a scoring function and a searching strategy [6] that will be now explained.

As we have previously seen, a $\mathbf{g}_t$ is a vector of features $\mathbf{g}_t = \{b_{1t}, \ldots, b_{|\mathcal{T}_{sp}|t}\}$ in the space $\mathcal{T}_{sp}$ and each document of the training set Ω_r can be represented as a vector $\mathbf{d}_m = (w_{1m}, \ldots, w_{|\mathcal{T}_{sp}|m})$ in the space $\mathcal{T}_{sp}$. A possible scoring function is the cosine similarity between these two vectors:

$$\mathcal{S}(\mathbf{g}_t, \mathbf{d}_m) = \frac{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt} \cdot w_{nm}}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt}^2} \cdot \sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} w_{nm}^2}} \tag{2.8}$$

and thus the optimization procedure would consist in searching for the best set of parameters Λ_t such that the cosine similarity is maximized $\forall \mathbf{d}_m$.

Therefore, the best $\mathbf{g}_t$ for the set of documents Ω_r is the one that produces the maximum score attainable for each document when used to rank Ω_r documents.

Since a score for each document $\mathbf{d}_m$ is obtained, we have:

$$\mathbf{S}_t = \{\mathcal{S}(\mathbf{g}_t, \mathbf{d}_1), \cdots, \mathcal{S}(\mathbf{g}_t, \mathbf{d}_{|\Omega_r|})\},$$

where each score depends on the specific set $\Lambda_t = (\tau, \boldsymbol{\mu})_t$.

To compute the best value of Λ we can maximize the score value for each document, which means that we are looking for the graph which best describes each document of the repository from which it has been learned. It should be noted that such an optimization maximizes at the same time all $|\Omega_r|$ elements of $\mathbf{S}_t$.

Alternatively, in order to reduce the number of the objectives being optimized, we can at the same time maximize the mean value of the scores and minimize their standard deviation, which turns a multi-objective problem into a two-objective one. Additionally, the latter problem can be reformulated by means of a linear combination of its objectives, thus obtaining a single objective function, i.e., *Fitness* ($\mathcal{F}$), which depends on Λ_t,

$$\mathcal{F}(\Lambda_t) = E\left[\mathbf{S}_t\right] - \sigma\left[\mathbf{S}_t\right],$$

where E is the mean value of all the elements of $\mathbf{S}_t$ and σ_m is the standard deviation. By summing up, the parameters learning procedure is represented as follows,

$$\Lambda^* = \underset{t}{\text{argmax}}\{\mathcal{F}(\Lambda_t)\}.$$

We will see next how the searching strategy phase has been conducted.

Since the space of possible solutions could grow exponentially, $|\mathcal{T}_{sp}| \leq 300$ [2] has been considered. Furthermore, the remaining space of possible solutions has been reduced by applying a clustering method, that is the *K-means* algorithm, to all ψ_{ij} and ρ_{is} values, so that the optimum solution can be exactly obtained after the exploration of the entire space.

[2] This number is usually employed in the case of Support Vector Machines.

2.3 From WWP to expanded query

As discussed before, a query expansion framework can be described as a modular system that essentially includes: a standard text retrieval module which, given a query (plain or expanded), returns a ranked set of documents out of an indexed corpus; a custom query expansion module which given a set of feedback documents, builds an expanded query to feed the text retrieval module. The proposed method can be summarized into three fundamental steps: initial user query and first search; user selection of relevant documents (minimal feedback); query expansion with subsequent search and retrieval of other relevant documents. In the first phase, user inputs a query to the text retrieval module, which performs an initial search and ranks results by using default similarity measures (for example Lucene standard *tf-idf*). During the second phase, the user checks the first pages of retrieved results and selects feedback documents. Once the optimal WWP structure has been extracted out of feedback documents (feature extraction phase), it has to be translated into an expanded query. This process, according to Fig.1.4, is called *query reformulation* and is carried out by considering a WWP graph (Fig.2.3) as a simple set of weighted word pairs (see *plain* WWP representation in Fig.2.4).

In fact, at this stage there's no more need to distinguish between aggregate roots and aggregates, although this hierarchical distinction was fundamental for the structure building process. Note that the *query reformulation* process is IR module dependant. There are several open source libraries providing full-text

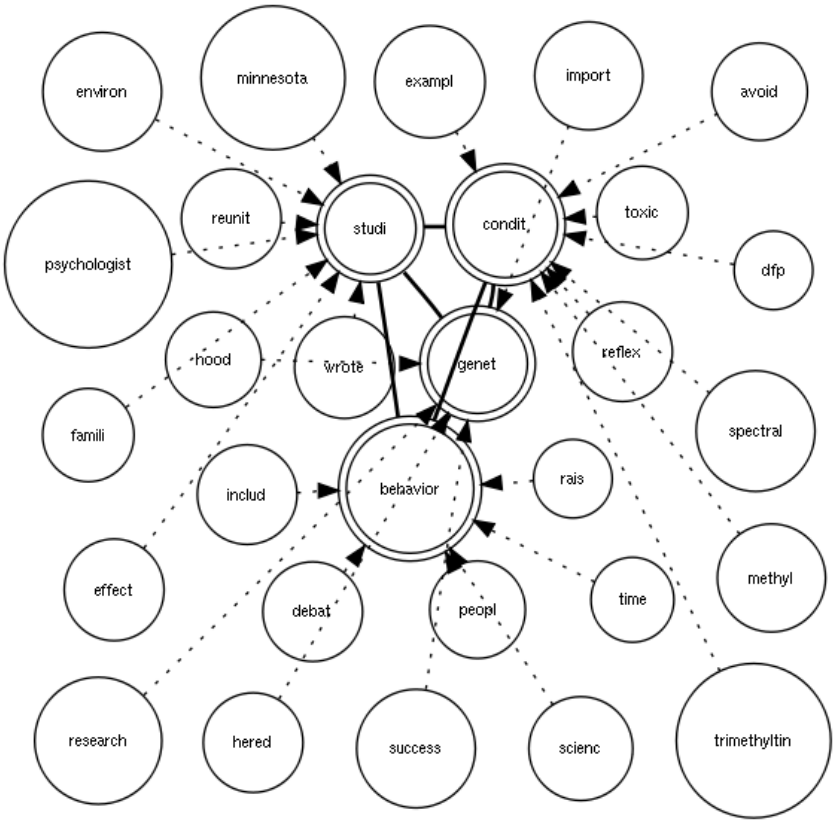


Figure 2.3 Example of a *Weighted Word Pairs* graph (Topic 402 TREC-8, "Behavioral genetics").

word1	word2	relation factor
condit	behavior	0.029258633
studi	behavior	0.05521098
genet	condit	0.019349499
genet	studi	0.021024829
genet	behavior	0.0048710075
studi	condit	0.026697233
includ	behavior	0.029998844
famili	studi	0.054471035

Figure 2.4 Fragment of plain WWP representation for the example in Fig. 2.3

search features among which Apache Lucene and Lemur Project [61] have been chosen since they handle complex query expansions through custom boolean weighted models. When using Lucene as IR [40] module, the WWP plain representation (Fig.2.4) is translated according to Lucene boolean model as follows:

```
(behavioral genetics)^1 OR (condit AND behavior)^0.02925
OR (studi AND behavior)^0.05521 ...
```

Every word pair is searched with a lucene *boost factor* chosen as the corresponding WWP relation factor, while the initial query is added with unitary boost factor (default).
When Lemur is used as IR module, WWP plain representation is translated into an expanded query using Indri query language. Inference networks, combined with language feature models, give

a solid theoretical basis for expressing information needs. In order to harness this model, Indri provides a query language that can express complex concepts [73]. The Indri query language is based on the successful Inquery structured query language. Both query languages are composed of operators, each of which can be considered a query node in an inference network. The Indri language contains the most popular operators from Inquery, along with many new operators that express concepts related to document structure. It also includes the window operators which allow the user to indicate that the location of query terms in a document affects relevance. The ordered window operator expresses that the terms should appear in a particular order in the document, while the unordered window operator merely requires terms to appear close together. Both operators have a distance parameter, N, that defines how close the terms need to be to each other. Indri also includes the *#combine* and *#weight* operators, which are similar in usage to the *#sum* and *#wsum* operators from Inquery. These terms allow users to combine beliefs from a variety of other query nodes effectively. Mathematically, the *#combine* operator corresponds to the *#and* operator from Inquery, while the *#weight* operator corresponds to the *#wand* operator proposed by Metzler.

Indri also incorporates the filter-require (*#filreq*) and filter-reject (*#filrej*) operators from Inquery, which are useful for filtering operations. The filter-require operator indicates that all relevant documents match a particular pattern; filter-reject indicates that relevant documents do not match a pattern.

In our case, we are mainly interested in belief operators from Lemur toolkit [61]. These allow to combine beliefs (scores) about

terms, phrases, etc. and can be both unweighted and weighted. With the weighted operators, weights can be assigned to certain expressions in order to control how much of an impact each expression within the query has on the final score. So if we choose to assign an equal impact to both the original query and the WWP graph, WWP plain representation can be formulated as follows:

```
#weight( 0.50 #combine(behavioral genetics)
0.50 #weight(0.02925 #band( condit behavior )
0.05521 #band( studi behavior ) ...
```

Here we recognize a weighted combination of original query and WWP graph. The weight "0.50" indicates that the same importance is given to the original query and the graph. The graph itself is translated as a weighted combination of "binary and" between word pairs where each weight corresponds to the WWP relation factor.
Next chapter shows results obtained from validating WWP approach with both IR modules in Text Retrieval and Text Categorization problems.

"I'm ... a rather simple person with a limited talent and perhaps a limited perspective."

BILL EVANS

Chapter 3

Experimental Results

WWP approach has been validated in two application fields: Query Expansion (in the domain of interactive text search engines) and Text Categorization. Standard datasets have been used for performance evaluation in both fields and results will be discussed in detail through the following sections.

3.1 WWP for Query Expansion

In Section 1.2.2 a modular query expansion framework has been introduced. Referring to such a scheme, performance comparison was carried out testing several FE/IR combinations. Two IR modules (Apache Lucene and Lemur) have been used for the evaluation, so that each of the following combination has been performed two times:

- **IR only** Unexpanded queries were performed. Only the first line of the chain in Figure 1.4 is involved, considering $\mathcal{RS} = (\mathbf{d}_1, \cdots, \mathbf{d}_N)$ as output. Results obtained for these cases are referred as *baseline*.

- **FE(WWP) + IR**. A WWP-based feature extraction (FE) method was used to expand the initial query and feed the IR module.

- **FE(KLD) + IR**. A Kullback Leibler Divergency [13] based feature extraction method was used to expand initial query and feed the IR module.

3.1.1 Datasets and Ranking Systems

The dataset from TREC-8 collections (minus the Congressional Record) was used for performance evaluation. It contains about 520,000 news documents on 50 topics (no.401-450) and relevance judgements for the topics. Figure 3.1 shows the number of relevant judged documents for each topic of the dataset. Word stopping and word stemming with single keyword indexing were performed. Query terms for each topic's initial search (baseline) were obtained by parsing the title field of a topic. For the baseline and for the first pass ranking (needed for feedback document selection) the default similarity measures provided by Lucene and Lemur has been used. Performance was measured with TREC's suggested evaluation measures, briefly discussed in next section: precision at different levels of retrieved results (P@5,10...1000), mean average precision (MAP), R-precision and binary preference (BPREF).

no.	Topic Title	# of Relevant Docs
401	foreign minorities, Germany	300
402	behavioral genetics	80
403	osteoporosis	21
404	Ireland, peace talks	142
405	cosmic events	38
406	Parkinson's disease	13
407	poaching, wildlife preserves	68
408	tropical storms	118
409	legal, Pan Am, 103	22
410	Schengen agreement	65
411	salvaging, shipwreck, treasure	27
412	airport security	123
413	steel production	69
414	Cuba, sugar, exports	39
415	drugs, Golden Triangle	136
416	Three Gorges Project	42
417	creativity	75
418	quilts, income	116
419	recycle, automobile tires	19
420	carbon monoxide poisoning	33
421	industrial waste disposal	83
422	art, stolen, forged	152
423	Milosevic, Mirjana Markovic	21
424	suicides	171
425	counterfeiting money	162
426	law enforcement, dogs	202
427	UV damage, eyes	50
428	declining birth rates	118
429	Legionnaires' disease	11
430	killer bee attacks	6
431	robotic technology	130
432	profiling, motorists, police	28
433	Greek, philosophy, stoicism	13
434	Estonia, economy	347
435	curbing population growth	117
436	railway accidents	180
437	deregulation, gas, electric	72
438	tourism, increase	173
439	inventions, scientific discoveries	219
440	child labor	54
441	Lyme disease	17
442	heroic acts	94
443	U.S., investment, Africa	102
444	supercritical fluids	17
445	women clergy	62
446	tourists, violence	162
447	Stirling engine	16
448	ship losses	46
449	antibiotics ineffectiveness	67
450	King Hussein, peace	293

Figure 3.1 Topics from TREC dataset with number of judged relevant documents available for each topic.

Evaluation measures

The two most frequent and basic measures for information retrieval effectiveness are *precision* and *recall*. *Precision (P)* is defined as the fraction of retrieved documents that are relevant:

$$P = \frac{\#(relevant\_items\_retrieved)}{\#(items\_retrieved)}$$

If we consider relevant retrieved items as *true positives (tp)* and non-relevant retrieved items as *false positives (fp)*, precision can be also written as follows:

$$P = \frac{tp}{tp + fp}$$

Recall (R) is the fraction of relevant documents that are retrieved:

$$R = \frac{\#(relevant\_items\_retrieved)}{\#(relevant\_items)}$$

In this case, if we refer to relevant not retrieved documents as *false negatives* and not retrieved non-relevant documents as *true negative*, the previous definition can be written as follows:

$$R = \frac{tp}{tp + fn}$$

The measures of precision and recall emphasize the return of true positives; they take into account what percentage of the relevant documents have been found and how many false positives have also been returned. These measures, typically computed using unordered sets of documents, can be extended to ranked results if evaluated for different sets of top k retrieved documents. For example, *Precision at k* (P@k) is computed after k documents

have been retrieved. In the TREC community the *Mean Average Precision* measure (MAP) is often used: it provides a single-figure measure of quality across recall levels with good discrimination and stability. Given a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved: in MAP this value is averaged over information needs. If the set of relevant documents for an information need $q_j \in Q$ is $\{d_1, \ldots d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k, then

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$

Since the number of relevant documents has a strong influence on precision at k, another measure called *R-Precision* is also used in TREC community. In order to compute R-precision it is required to know a set of relevant document *Rel* and calculate the precision of the top *Rel* documents returned. This measure de-emphasizes the exact ranking of the retrieved relevant document, which is useful in cases where large numbers of documents are available (TREC). The average R-Precision for a run can be computed by taking the mean of the R-Precisions of individual topics in the run. For example, we can assume a run consisting of two topics, one with 50 relevant documents and another with 10 relevant documents. If the retrieval system returns 17 relevant documents in the top 50 documents for the first topic, and 7 relevant documents in the top 10 for the second topic, then the run's R-Precision would be $\frac{\frac{17}{50}+\frac{7}{10}}{2} = 0.52$. The *Binary Pref-*

erence (bpref) measure has been introduced for situations where relevance judgments are not complete. It computes a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant documents. Thus, it emphasizes the relative ranks of judged documents only. The bpref measure can be defined as follows:

$$bpref = \frac{1}{R}\sum_{r}(1 - \frac{|n \text{ ranked higher than } r|}{min(R, N)})$$

where R is the number of judged relevant documents, N is the number of judged irrelevant documents, r is a relevant retrieved document and n is a member of the first R irrelevant retrieved documents.

All the evaluation measures on TREC dataset were computed using the *trec_eval* program written by Chris Buckley [10].

3.1.2 Parameter Tuning

The two most important parameters involved in the computation of WWP, given the number of documents for training, are the *number of aggregate roots H* and the *number of pairs*. The number of aggregate roots can be chosen as a trade off between retrieval performances and computational times (see Fig.3.2); our choice was $H = 4$ since it seemed to be the best compromise (about 6 seconds per topic)[1]. However, we want to emphasize method effectiveness more than algorithm efficiency since algorithm coding has not been completely optimized yet.

[1]Results were obtained using an *Intel Core 2 Duo 2,40 GHz* PC with *4GB RAM* with no other process running.

H	MAP(%)	P@5(%)	Time (s)
2	26,00	72,00	3,98
3	27,95	73,60	4,6
4	29,09	76,00	6,06
5	29,17	76,24	9,5
6	30,04	73,60	12,04

Figure 3.2 The number of aggregate roots H can be chosen as a trade off between retrieval performances and computational times. Our choice was $H = 4$.

Fig.3.3 shows results of baseline and WWP method when changing *number of pairs* from 20 to 100 where the number of documents is fixed to 3: in this analysis, Lucene IR module is used. According to the graph, our system always provides better performances than baseline; the change in number of pairs has a great impact especially on precision at 5 where 60 pairs achieve the best results. Anyway, if we consider precision at higher levels together with map values, 50 pairs seem to be a better choice also for shorter computational times. Fig.3.4 shows results of baseline and WWP method when changing *number of training documents* (Lucene IR Module used): here we can see that the overall behaviour of the system is better when choosing 3 relevant documents for training. Once again the system outperforms baseline especially at low precision levels.

Since a relevance feedback based method is proposed, it was useful to analyze *precision@10* evaluated for each topic in response to the base query. Such an analysis, performed using both Lucene

(Fig.3.5) and Lemur (Fig.3.6) as IR modules, allows to check how many relevant documents are retrieved for each topic in the first ten results. For each IR module, we can summarize outcomes as follows:

- **Lucene IR**. In the first ten results, there's no set of 2 relevant documents for 12 topics and no set of 3 relevant documents for 19 topics (see Figure 3.5)
- **Lemur IR**. In the first ten results, there's no set of 2 relevant documents for 7 topics and no set of 3 relevant documents for 13 topics (as shown in Fig. 3.6).

Therefore, a pseudo relevance approach for this dataset could be not recommended when Lucene and Lemur are used as IR modules, since performance is compromised by the lack of relevant sets in top k retrieved for certain topics. Discussed analysis led to choose the following settings for the experimental stage: 4 aggregate roots, 50 pairs, 3 training documents.

3.1.3 Comparison with other methods

Figure 3.7 shows a first performance comparison between WWP method, baseline (unexpanded query) and a random weighted WWP when using both Lucene and Lemur as IR modules. A random weighted WWP is obtained by "corrupting" the building process of a standard WWP structure so that random probabilities are considered instead of those coming from LDA computation; this allows to check the strength of the method in both term extraction and relations learning. In Fig.3.8 WWP method is compared with

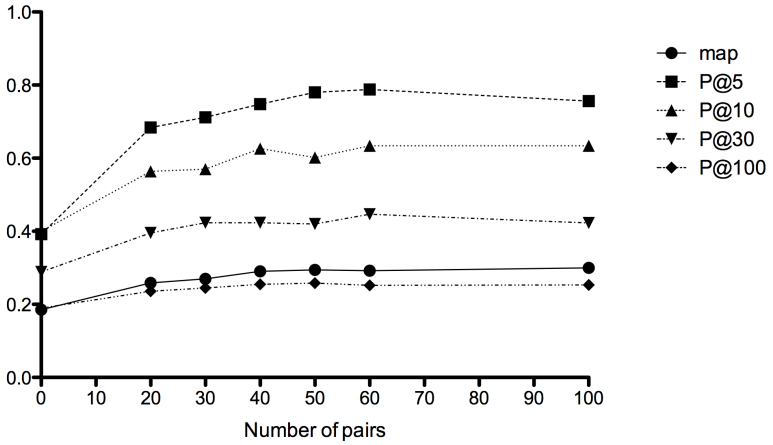


Figure 3.3 *WWP* performance when changing number of pairs

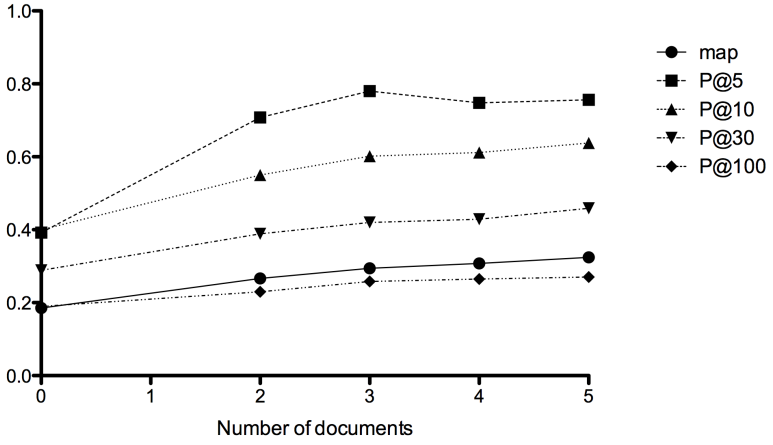


Figure 3.4 *WWP* performance when changing number of training documents

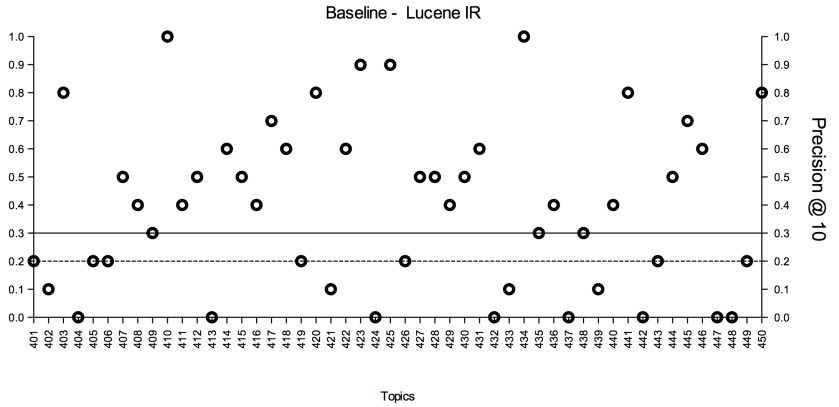


Figure 3.5 Precision@10 analysis of baseline results obtained for each topic with Lucene IR

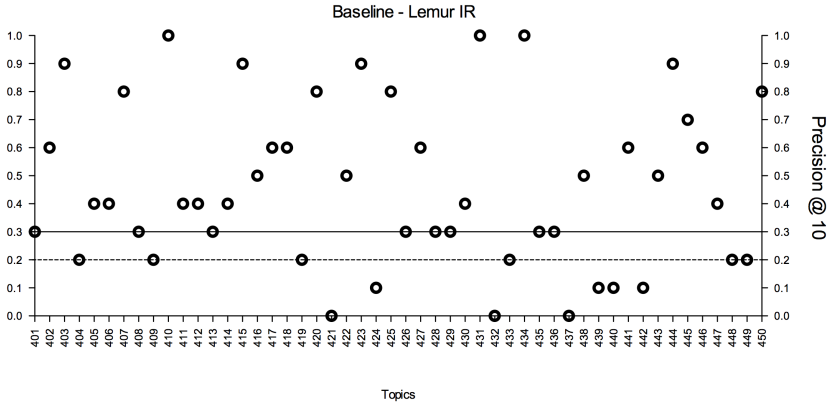


Figure 3.6 Precision@10 analysis of baseline results obtained for each topic with Lemur IR

IR	**Lucene**		**Lemur**	
FE	WWP	WWP(rand)	WWP	WWP(rand)
relret	3068	1472	3285	1577
map	0,2909	0,1236	0,3069	0,1319
Rprec	0,3265	0,1625	0,3324	0,1628
bpref	0,3099	0,1985	0,3105	0,1987
P@5	0,76	0,564	0,736	0,5446
P@10	0,602	0,404	0,58	0,3886
P@100	0,2612	0,1202	0,2562	0,1178
P@1000	0,0614	0,0294	0,0657	0,0315

Figure 3.7 Results comparison for *WWP* against random weighted WWP with 3 training documents.

baseline and *Kullback-Leibler* divergence based method [13]. Here we see that WWP outscores KLD, random weighted WWP and baseline especially for low level precision while having good performances for other measures. However these results are obtained without removing feedback documents from the dataset, which is a common behaviour for text retrieval systems. One could argue that a big improvement in low level precision is essentially due to feedback documents being better ranked for the use of WWP. Therefore, another performance evaluation was carried out using only the residual collection (RSD) where feedback documents are removed. Results for this evaluation are shown in Fig.3.9: here we see that WWP method provides better global performance also with residual collection.

IR	**Lucene**			**Lemur**		
FE	-	KLD	WWP	-	KLD	WWP
relret	2267	2304	3068	2780	2820	3285
map	0,1856	0,1909	0,2909	0,2447	0,2560	0,3069
Rprec	0,2429	0,2210	0,3265	0,2892	0,2939	0,3324
bpref	0,2128	0,2078	0,3099	0,2512	0,2566	0,3105
P@5	0,3920	0,5200	0,7600	0,4760	0,5720	0,7360
P@10	0,4000	0,4300	0,6020	0,4580	0,4820	0,5800
P@100	0,1900	0,1744	0,2612	0,2166	0,2256	0,2562
P@1000	0,0453	0,0461	0,0614	0,0556	0,0564	0,0657

Figure 3.8 Results comparison for unexpanded query, KLD and WWP (FE) using Lucene and Lemur as IR modules.

Since we are interested in finding out strengths and weaknesses of the proposed approach, a detailed analysis of Average Precision, Binary Preference and Precision@10 for each topic has been also reported in this dissertation. Figures 3.10 and 3.11 show an Average Precision analysis for each topic when using Lucene and Lemur IR respectively. The bar charts (which are not stacked!) allow higher values to hide lower ones so that we can easily identify cases where WWP performs worse than KLD and/or baseline. For example, Figure 3.10 shows that the use of an unexpanded query on Lucene for topic 423 achieve better average precision performance than both KLD and WWP; the use of expanded terms in this case seems to introduce noise in the retrieval task. A comparative analysis shows how different IR modules can react to the

IR	**Lucene**			**Lemur**		
FE	-	KLD	WWP	-	KLD	WWP
relret	2117	2178	2921	2630	2668	3143
map	0,1241	0,1423	0,2013	0,1861	0,1914	0,2268
Rprec	0,1862	0,1850	0,2665	0,2442	0,2454	0,2825
bpref	0,1546	0,1716	0,2404	0,1997	0,2044	0,2471
P@5	0,2360	0,3920	0,4840	0,3880	0,4120	0,5120
P@10	0,2580	0,3520	0,4380	0,3840	0,3800	0,4560
P@100	0,1652	0,1590	0,2370	0,1966	0,2056	0,2346
P@1000	0,0423	0,0436	0,0584	0,0526	0,0534	0,0629

Figure 3.9 Results comparison for unexpanded query, KLD and WWP using Lucene or Lemur with RSD.

same problem. Note that, when Lemur is used (Fig. 3.11), WWP is able to achieve the best performance in terms of average precision for topic 423 but here we find some issues for topic 430. Figures 3.12 and 3.13 show a similar analysis for Binary Preference. Also in this case, we find that both WWP and KLD have poor performance for topic 423 when Lucene is used (Fig. 3.12) and the topic 430 reveals a performance issue too. Anyway, the use of Lemur allows WWP to obtain the best performance on topic 423, but KLD is still to prefer for topic 430. Another interesting comparative analysis has been reported for Precision@10. Figure 3.14 shows that, when Lucene is used, the use of expansion terms compromise performances for some topics; in particular for topics 423, 425, 441 baseline performs significantly better than WWP

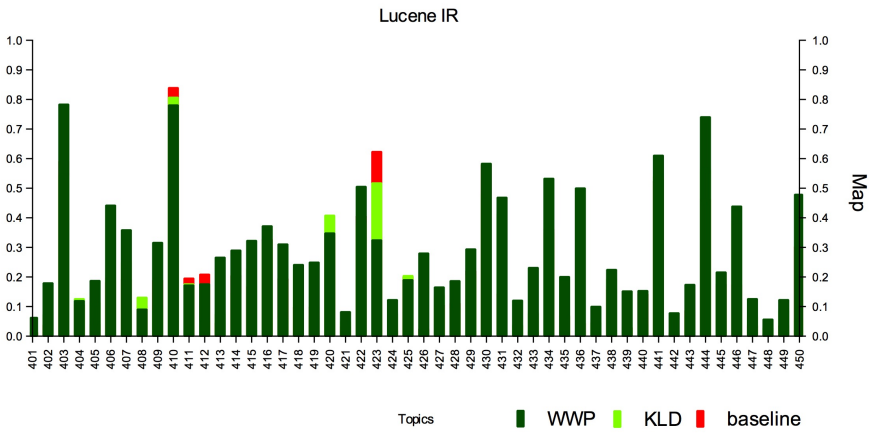


Figure 3.10 MAP analysis of WWP, KLD, and baseline for each topic with Lucene IR

and KLD. However, when Lemur is used, we see that WWP gives better results for most topics in terms of Precision@10.

As seen before, in some cases the use of query expansion can have the drawback of lowering performances. Such a behaviour can be due to different factors, so that some considerations need to be made. First of all, if we check the number of judged relevant documents available for each topic (Fig. 3.1), we realize that, for certain topics, such a number is very small compared to the size of the whole dataset. Moreover, not every document in the considered dataset has been judged and there is a large subset of negative examples (document judged as non relevant) which our system doesn't take into account to build the graph. Finally, a deep analysis of the documents selected for graph building could reveal the presence of bad formed periods or inconsistent document structures. Since such an analysis could be very subjective, this aspect has not been considered for the evaluation.

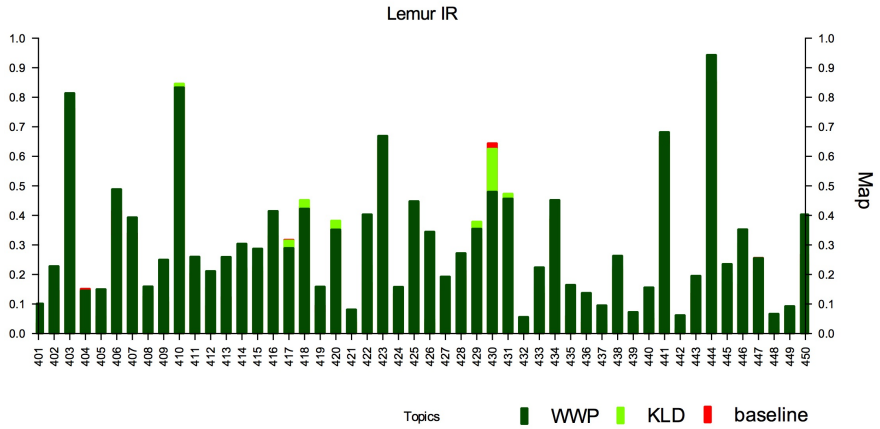


Figure 3.11 MAP analysis of WWP, KLD, and baseline for each topic with Lemur

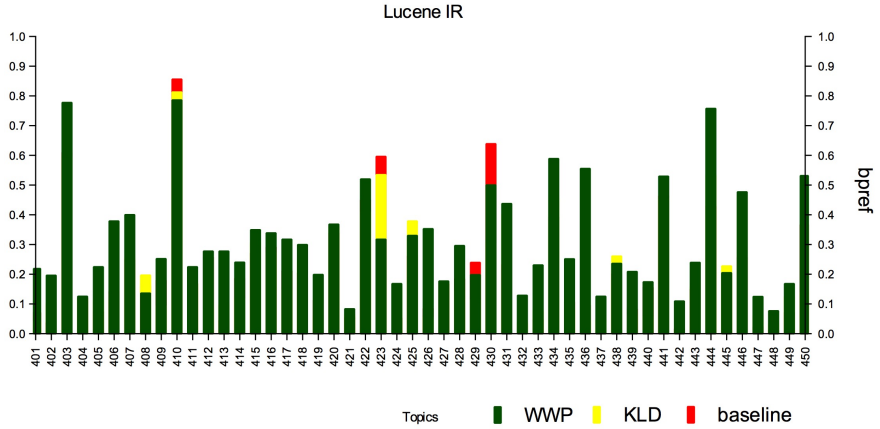


Figure 3.12 Binary Preference analysis of WWP, KLD, and baseline for each topic with Lucene IR

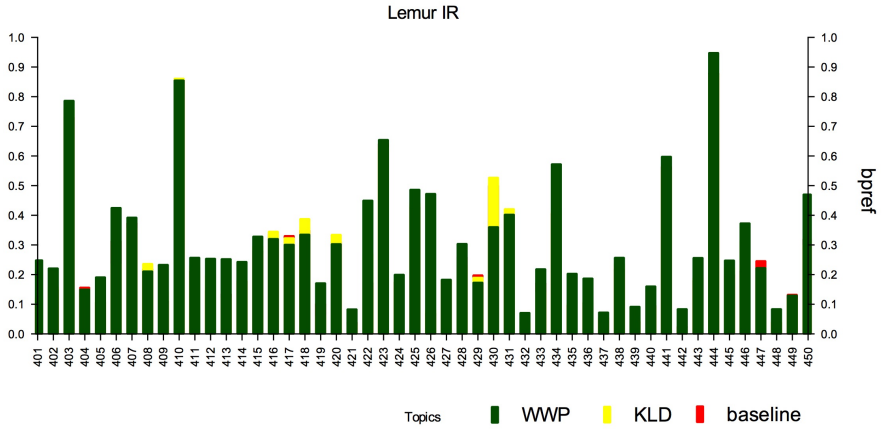


Figure 3.13 Binary Preference analysis of WWP, KLD, and baseline for each topic with Lemur IR

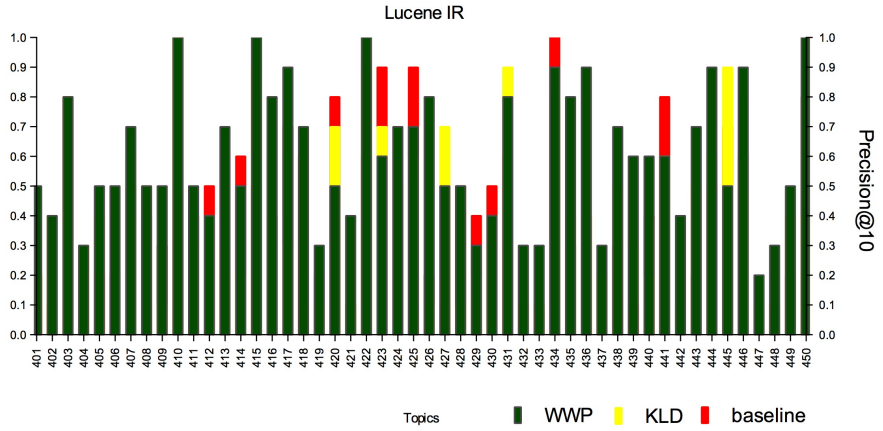


Figure 3.14 Precision@10 analysis of WWP, KLD, and baseline for each topic with Lucene IR

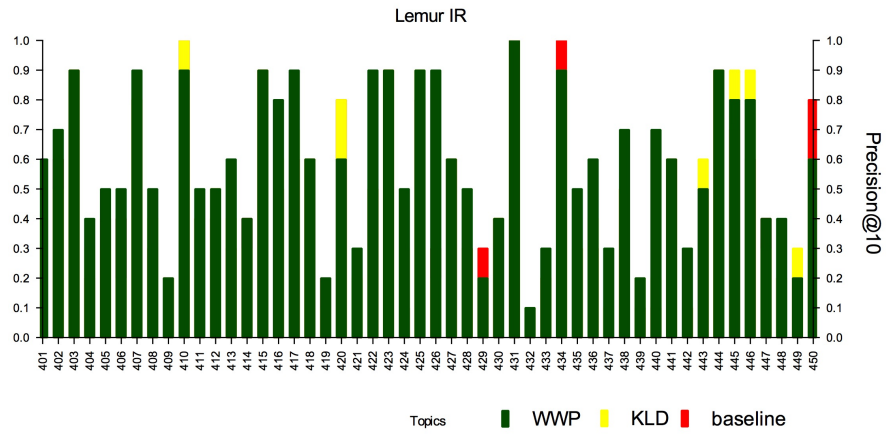


Figure 3.15 Precision@10 analysis of baseline results obtained for each topic with Lemur IR

3.2 WWP for Text Categorization

As discussed before, a binary classifier, also known as *hard* classifier, is mostly used in text categorization problems where the task is to assign each document to each class. Unfortunately, the proposed approach behaves more as a document-ranking text classifier, namely a *soft* decision based classifier. In fact, once a WWP structure has been learnt from the training set for a class, that structure is used to query the test set through a IR module obtaining a set of ranked documents as output.

A way to turn a soft classifier into a hard one is to define a threshold γ_i such that the $CSV_i(\mathbf{d}_m) \geq \gamma_i$ (*Categorization Status Value*) is interpreted as T while $CSV_i(\mathbf{d}_m) \leq \gamma_i$ is interpreted as F. An experimental method, known as CSV *thresholding* [70] has been adopted and consists in testing different values for γ_i on a subset of the training set (the *validation* set) and choosing the value which maximizes effectiveness.

3.2.1 Datasets and Ranking Systems

Performance evaluation was carried out using Reuters-21578 repository which is a collection of 21,578 newswire articles, originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd.. Classes from a set of 118 topic categories have been associated to each article: a document may belong to several classes or none, but the commonest case is a single assignment (documents with at least one class received an average of 1.24 classes).

For this task the ModApte split has been used: it includes only documents that were viewed and assessed by a human indexer, and comprises 9,603 training documents and 3,299 test documents. The distribution of documents in classes is very uneven so evaluation was conducted using only documents belonging to the 10 largest classes [18][2].

Note that WWP structure is more complex than a simple list of keywords since it takes into account relations between terms and the hierarchical differentiation between aggregate words and aggregate roots. To demonstrate the discriminative property of such features and justify the overhead in complexity coming from their introduction, we have to prove that results obtained with the proposed approach are significantly better than results obtained when the same classification task is performed using a simple list of weighted words.

In single label or binary classification we usually have a training set containing examples that are labeled as $\mathbf{c}_i$ or $\overline{\mathbf{c}}_i$. A typical

[2]Note that considering the 10 largest classes means 75% of the training set and 68% of the test set.

classifier learns from both sets of examples and is capable to assign a new document to the category $\mathbf{c}_i$ or $\overline{\mathbf{c}}_i$.

But a standard WWP structure is learned only from documents labeled as $\mathbf{c}_i$ (positive examples) so documents belonging to the category $\overline{\mathbf{c}}_i$ are not used: this would make the proposed approach not directly comparable with existing methods. Anyway, a comparison has been performed with linear Support Vector Machines (SVM) learned on the same percentage of the training set (both positive and negative examples) with a mutual information based term selection.

The aim of the evaluation phase is twofold: to demonstrate the discriminative property of WWP compared with a simpler method using the same keywords selected by WWP but neglecting relations (named the Words List); to demonstrate that WWP achieves a good performance when 1.4% of the training set is employed for each class. The *any-of problem* and 10 two-class classifiers, one for each class, have been considered where a two-class classifier for class $\mathbf{c}_i$ is the classifier for the class $\mathbf{c}$ and its complement $\overline{\mathbf{c}}_i$. Defining tp_i as true positive, tn_i as true negative, fp_i as false positive and fn_i as false negative for the category $\mathbf{c}_i$ ([70, 18]), for each classifier several measures were evaluated: precision and recall; micro-average precision and recall; F_1 measure; micro-average F_1; macro-average F_1. Some details about such measures will be discussed in the next section.

Evaluation measures

In order to evaluate classification performances for each category c_i, *precision* and *recall* were computed following the definitions

previously provided: $P_i = \frac{tp_i}{tp_i+fp_i}$ and $R_i = \frac{tp_i}{tp_i+fn_i}$.
A single measure that trades off precision versus recall was also considered, the *F-measure*. It is defined as the weighted harmonic mean of precision (P) and recall (R):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

where $\beta^2 = \frac{1-\alpha}{\alpha}$. Since $\alpha \in [0,1]$, we have that $\beta^2 \in [0,\infty]$. For a balanced *F-measure*, where precision and recall are equally weighted, $\alpha = 1/2$ or $\beta = 1$ are often chosen obtaining the common F_1 measure. For a given category c_i:

$$F_{1i} = 2 \cdot \frac{P_i \cdot R_i}{P_i + R_i}$$

Note that *global* measures are usually required to properly tune and evaluate a text categorization system. Such methods makes use of *micro-averaging* and *macro-averaging*. Micro-averaged values are calculated by constructing a global contingency table (for true/false positives and true/false negatives) and calculating precision, recall and f-measure using those sums:

$$P_{micro} = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} tp_i + fp_i}$$

$$R_{micro} = \frac{\sum_{i=1}^{|C|} tp_i}{\sum_{i=1}^{|C|} tp_i + fn_i}$$

$$F_{1micro} = 2 \cdot \frac{P_{micro} \cdot R_{micro}}{P_{micro} + R_{micro}}$$

where $|C|$ is the number of categories. Macro-averaged scores are obtained by first calculating precision and recall for each category and then taking the average of these:

$$P_{macro} = \sum_{i=1}^{|C|} P_i$$

$$R_{macro} = \sum_{i=1}^{|C|} R_i$$

$$F_{1macro} = \frac{1}{|C|} \sum_{i=1}^{|C|} F_{1i}$$

The main difference between these two methods is that micro-averaging gives equal weight to every document (it is also called a *document-pivoted* measure) while macro-averaging gives equal weight to every category (*category-pivoted* measure).

3.2.2 Parameter Tuning

The threshold γ for the categorization status value has been set by evaluating aggregate measures: micro-precision, micro-recall and micro F1 (see Fig. 3.16); $\gamma = 0.1$ was the choice for all the topics. After tuning the classifier, macro-F_1 measure was evaluated for different sizes of the reduced training set Υ_r. Since document lenght can vary strongly across different documents of the dataset, every subset was extracted from the corresponding training set as a fraction of its size in KB. Fig.3.17 shows the behavior of the classifier: there's a degradation of performance as the size of the training set increases. This suggests that WWP becomes less discriminative as the number of labeled examples increases. For this reason, Υ_r was chosen as about 1.4% of Ω_r. In Fig. 3.18 is reported the comparison between the dimension of Υ_r and the original training set Ω_r. The selection was performed 100

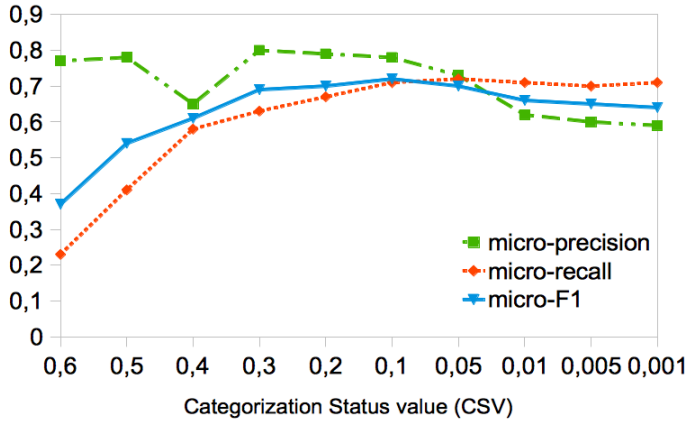


Figure 3.16 Tuning of the threshold for γ.

times in order to make the results independent from the particular document selection. As a result, 100 repositories were available and 100 WWP structures were extracted out of them performing the parameters learning described above.

3.2.3 Comparison with other methods

Since each optimization procedure leads to a different WWP structure, we have a different number of pairs for each structure. The average number of pairs for each topic and the corresponding average number of terms have been calculated. Note that the average size of $|\mathcal{T}_{sp}|$ is 116, while the average size of $|\mathcal{T}_s|$ is 33. The overall number of features used by our method is, independently of the topic, less than the number considered in the case of Support Vector Machines where the term selection process results in $|\mathcal{T}|_s = 300$. In Figure 3.18 F_1 measure, micro-F_1 and macro-F_1 obtained by the WWP graph $\mathbf{g}$, word list $\mathbf{w}$ and support vector machines (SVM). The best values and the average values obtained

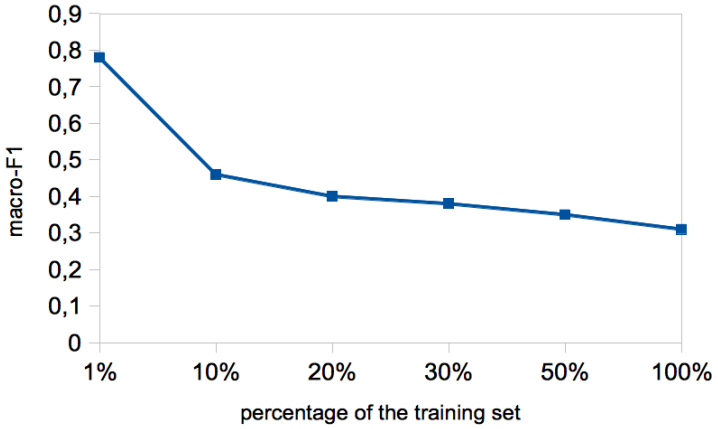


Figure 3.17 Different values of macro-F_1 for different percentages of the training set Υ_r.

by performing the classification of all 100 examples of the reduced training set has been reported. It is surprising how the proposed method, even if the training set is smaller than the original one, is capable of classifying in most cases with an accuracy sometimes comparable and mostly better than Support Vector Machines. Note that the performance of the proposed method is, independently of the topic, better than the simple word list, so demonstrating that WWP representation has a better discriminative power. Finally, it should be noticed that even if good performances are obtained when a simple word list is employed, such a list is however composed of WWP terms. This demonstrates that WWP could be useful also to select the most discriminative words from the space Υ_s.

Topic	Ω_r(KB)	Υ_r(KB)		**g@max**	**g@av**	**w@max**	**w@av**	**SVM@max**	**SVM@av**		
earn	957	14		91	76	82	69	95	66		$\Downarrow$
acq	902	13		63	44	53	38	63	35		$\Leftrightarrow$
money-fx	476	7		46	30	39	23	37	09		$\Uparrow$
grain	359	5		66	40	54	35	48	04		$\Uparrow$
crude	356	5		70	42	60	40	47	10		$\Uparrow$
trade	440	6		58	42	43	39	27	06		$\Uparrow$
interest	267	4		50	34	38	24	09	01		$\Uparrow$
ship	137	2		68	18	59	12	16	01		$\Uparrow$
wheat	229	3		86	43	72	31	26	02		$\Uparrow$
corn	153	2		65	23	54	16	10	02		$\Uparrow$
		F_{1micro}		**66**	**39**	**46**	**23**	**38**	**14**		$\Uparrow$
		F_{1macro}		**74**	**53**	**56**	**33**	**61**	**28**		$\Uparrow$

Figure 3.18 Average dimension of the reduced training set Υ_r and original dimension of Ω_r. F_1 measure, F_{1micro} and F_{1macro} for the graph (**g**), word list (**w**) and Support Vector Machines (SVM). The arrows column shows the increment of **g** performance compared with other methods.

"...and incidentally, I am not self-taught. Everybody who has given me a moment of beauty, significance or excitement has been a teacher."

GEORGE RUSSELL

Chapter 4

Conclusions and future works

In this dissertation an alternative method for supervised text classification, relying on a Weighted Word Pairs (WWP) structure, has been presented. Such a method, which has shown to be effective when small training sets are available, was validated in both Text Retrieval (Query Expansion) and Text Categorization fields using standard datasets.

The experimental phase for Query Expansion field was conducted with the use of TREC-8 dataset, containing approximately 520 thousand pre-classified documents. A performance comparison between the baseline (results obtained with no expanded query), WWP structure and query expansion method based on the Kullback Leibler divergence was carried out by computing typical information retrieval measurement: precision at various levels,

mean average precision, binary preference, R-precision. Results are very encouraging: Weighted Word Pairs hierarchical representation seems capable of retrieving a greater number of relevant documents than a less complex representation based on a list of weighted words (KLD). Then, it can be employed in all those text mining tasks that consider matching between patterns represented as textual information as well as in sentiment analysis and detection tasks. Note that the proposed approach computes the expanded queries considering only endogenous knowledge. It is well known that the use of external knowledge, for instance WordNet, could clearly improve the accuracy of information retrieval systems and this integration for WWP could be considered an interesting future work. WWP structure has been also validated in the context of document categorization. In this case, the structure WWP combined with a module of Information Retrieval has been used to implement a document-ranking text classifier, which is able to make a soft decision: it draws up a ranking of documents that requires the choice of an appropriate threshold (Categorization Status Value) in order to obtain a binary classification. This threshold was chosen by evaluating performance on a *validation set* in terms of micro-precision, micro-recall and micro-F1. The dataset Reuters-21578, consisting of about 21 thousand newspaper articles, has been used; in particular, evaluation was performed on the ModApte split (10 categories), which includes only documents classified manually by humans . The experiment was carried out by selecting the 1% randomly in the training set for each category and this selection was made 100 times so that the results are not biased by the specific subset. The performance, evaluated

by calculating the F1 measure (harmonic mean of precision and recall), was compared with the Support Vector Machines, in the literature referred as the state of the art in the classification of such a dataset. The results show that when the training set is reduced to 1%, the performance of the classifier based on WWP are on average higher than those of SVM. Note that most text classifiers are trained using both positive and negative examples. An interesting future work could be the use of two WWP structures (positive and negative) to observe if there is any improvement in classification performance. Since a WWP-based classifier is a document ranking classifier, the combined use of two WWP structure requires a proper choice of the categorization status value. For example, given the document rankings for each WWP (positive and negative), scores of documents appearing in both cases, could be properly combined to refine results.

Another challenging work, which is in progress for the author, is the use of a WWP graph as a starting point to build a probabilistic terminological ontology out of a corpus of documents. Note that ontology learning from text is the process of identifying terms, concepts, relations and axioms from textual information and using them to construct and maintain ontology [77]. It can be viewed essentially as the process of deriving high level *concepts* and *relations* as well as *axioms* from information to form an ontology. It's like a problem of reverse engineering: the author of a document or a text has a domain model in her mind which shares with other authors writing texts about the same domain. The ontology learning from text is generally composed by five phases that aim to return five outputs: terms, concepts, tax-

onomic relations, non-taxonomic relations and axioms [11]. To obtain each output, some tasks have to be accomplished and the techniques employed for each task may change among systems. Starting from the terms is possible to derive the concepts that can be formed by grouping similar terms and labeling them. The grouping phase involves discovering the variants of a term and grouping them together, while the concept's label can be inferred by the use of existing background knowledge, such as WordNet, that may be used to find the name of the nearest common ancestor. The relations model the interactions among the concepts in ontology: in general, two types of relations can be recognized in ontology: taxonomic and non-taxonomic relations. Taxonomic relations, that are hypernym, build hierarchies and can be labeled as "is-a" relations [20]. This kind of relations can be performed in various ways such as using predefined relations from existing background knowledge, using statistical subsumption models, relying on semantic similarity between concepts and utilizing linguistic and logical rules or patterns. The non-taxonomic relations are the interactions among the concepts other than hypernymy and their extraction is a challenging task. In this context verbs play a significant role such as the support of domain experts. Axioms are propositions or sentences that are always taken as true and are the starting point for deducing other truth, verifying the correctness of the ontological elements and defining constraints. The process of learning axioms is still complex and there are few examples in literature. That said, a local terminological ontology can be build out of a WWP graph as a taxonomy of discovered concepts for a single topic and for a given aggregate root node with a set

of generic relationships among the root node and other semantic nodes, using *WordNet* as general lexical vocabulary. A possible approach, which will be developed as a future work, could be this: determine common hypernyms between the aggregate root node and aggregated words; add hypernyms to the ontology as semantic nodes if they are semantically similar to the root node; update ontology by computing the correct IS_A relationships among the concepts, corresponding to the ancestor and leave nodes. Some research activities in this field are already being carried out by the author's research group. More in general, the use of both statistical and semantic techniques allows to obtain effective domain ontologies particularly suitable for a number of applications such as topic detection and tracking, opinion and sentiment analysis, text mining and classification and so on.

Acknowledgements

First I'd like to thank God, especially for giving me the power to believe in my passions and pursue my dreams. I would also like to extend my deepest gratitude to my family for all the love and support: my parents for believing in me and encouraging my studies over the years; my brother for the advices; Alma for being always there and sharing goals and concerns.
A very special thank to my advisor, Massimo De Santo, and to Francesco Colace and Paolo Napoletano who patiently assisted me in making this work. I'm also grateful to everyone at NCLab (NITe) for their support: Angelo Marcelli, Antonio Della Cioppa, Rosa Senatore, Adolfo Santoro and Antonio Parziale. Many thanks to Antonio Picariello and Vincenzo Moscato from UniNA for the collaboration and the advices. Last but not least, thanks to all my friends, to the Music and to everyone "who has given me a moment of beauty, significance or excitement".

Luca Greco

Bibliography

[1] R.o. duda, p.e. hart, and d.g. stork, pattern classification, new york: John wiley & sons, 2001, pp. xx + 654, isbn: 0-471-05669-3. *J. Classif.*, 24(2):305–307, September 2007.

[2] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 163–222. Springer, 2012.

[3] Chidanand Apte, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251, July 1994.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, New York, 1999.

[5] J. Bhogal, A. Macfarlane, and P. Smith. A review of ontology based query expansion. *Information Processing & Management*, 43(4):866 – 886, 2007.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[7] D. M Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(993–1022), 2003.

[8] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. new edition.

[10] Buckley. trec_eval evaluation program, August 2008.

[11] Paul Buitelaar and Bernardo Magnini. Ontology learning from text: An overview. In *In Paul Buitelaar, P., Cimiano, P., Magnini B. (Eds.), Ontology Learning from Text: Methods, Applications and Evaluation*, pages 3–12. IOS Press, 2005.

[12] James Callan, W. Bruce Croft, and Stephen M. Harding. The inquery retrieval system. In *In Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83. Springer-Verlag, 1992.

[13] Claudio Carpineto, Renato de Mori, Giovanni Romano, and Brigitte Bigi. An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.*, 19:1–27, January 2001.

[14] Soumen Chakrabarti, Byron Dom, , Piotr Indyk, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks, 1998.

[15] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *In Proceedings of the 23rd VLDB Conference*, 1997.

[16] Soumen Chakrabarti, Shourya Roy, and Mahesh Soundalgekar. Fast and accurate text classification via multiple linear discriminant projections, 2002.

[17] Sutanu Chakraborti, Rahman Mukras, Robert Lothian, Nirmalie Wiratunga, Stuart Watt, and David Harper. Supervised latent semantic indexing using adaptive sprinkling. In *In IJCAI*, 2007.

[18] Prabhakar Raghavan Christopher D. Manning and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University, 2008.

[19] Kenneth W. Church and William A. Gale. Poisson mixtures. *Natural Language Engineering*, 1:163–190, 1995.

[20] Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning taxonomic relations from heterogeneous evidence.

[21] Fabio Clarizia, Luca Greco, and Paolo Napoletano. A new technique for identification of relevant web pages in informational queries results. In *Proceedings of the 12th International*

Conference on Enterprise Information Systems: Databases and Information Systems Integration, pages 70–79, 8-12 June 2010.

[22] Fabio Clarizia, Luca Greco, and Paolo Napoletano. An adaptive optimisation method for automatic lightweight ontology extractions. In J. Filipe and J. Cordeiro, editors, *Lecture Notes in Business Information Processing*, pages 357–371. Springer-Verlag Berlin Heidelberg, 2011.

[23] William Cohen and Haym Hirsh. Joins that generalize: Text classification using whirl. In *In Proc. of the Fourth Int Conference on Knowledge Discovery and Data Mining*, pages 169–173, 1998.

[24] William W. Cohen. Learning trees and rules with set-valued features. pages 709–716, 1996.

[25] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *ACM Transactions on Information Systems*, pages 307–315. ACM Press, 1996.

[26] Kevyn Collins-Thompson and Jamie Callan. Query expansion using random walk models. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 704–711, New York, NY, USA, 2005. ACM.

[27] William S. Cooper. Some inconsistencies and misnomers in probabilistic information retrieval. In *Proceedings of the 14th*

annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '91, pages 57–61, New York, NY, USA, 1991. ACM.

[28] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[29] Steven P. Crain, Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Dimensionality reduction and topic modeling: From latent semantic indexing to latent dirichlet allocation and beyond. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 129–161. Springer, 2012.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[31] Pedro Domingos, Michael Pazzani, and Gregory Provan. On the optimality of the simple bayesian classifier under zero-one loss. In *Machine Learning*, pages 103–130, 1997.

[32] H. Drucker, Donghui Wu, and V.N. Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048 –1054, sep 1999.

[33] Susan Dumais. Hierarchical classification of web content. pages 256–263. ACM Press, 2000.

[34] Susan Dumais, Thorsten Joachims, Krishna Bharat, and Andreas Weigend. SIGIR 2003 workshop report: implicit

measures of user interests and preferences. *SIGIR Forum*, 37(2):50–54, 2003.

[35] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, CIKM '98, pages 148–155, New York, NY, USA, 1998. ACM.

[36] Georges Dupret. Latent concepts and the number orthogonal factors in latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 221–226, New York, NY, USA, 2003. ACM.

[37] Efthimis N. Efthimiadis. Query expansion. In Martha E. Williams, editor, *Annual Review of Information Systems and Technology*, pages 121–187. 1996.

[38] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[39] Imola Fodor. A survey of dimension reduction techniques. Technical report, 2002.

[40] Apache Software Foundation. Apache lucene - scoring, 2011. letzter Zugriff: 20. Oktober 2011.

[41] T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum. Topics in semantic representation. *Psychological Review*, 114(2):211–244, 2007.

[42] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the Twenty-Second Annual International SIGIR Conference*, 1999.

[43] Thomas Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI99*, pages 289–296, 1999.

[44] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):995 –1006, aug. 2004.

[45] Peg Howland, Moongu Jeon, and Haesun Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25:165–179, 2003.

[46] Mike James. *Classification Algorithms*. John Wiley, 1985.

[47] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing & Management*, 44(3):1251 – 1266, 2008.

[48] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing & Management*, 36(2):207–227, 2000.

[49] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. pages 143–151, 1997.

[50] Thorsten Joachims. Transductive inference for text classification using support vector machines. pages 200–209. Morgan Kaufmann, 1999.

[51] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 128–136, New York, NY, USA, 2001. ACM.

[52] Youngjoong Ko and Jungyun Seo. Text classification from unlabeled documents with bootstrapping and feature projection techniques. *Inf. Process. Manage.*, 45:70–83, January 2009.

[53] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 170–178, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[54] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 81–89, New York, NY, USA, 1998. ACM.

[55] Hao Lang, Donald Metzler, Bin Wang, and Jin-Tao Li. Improved latent concept expansion using hierarchical markov random fields. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 249–258, New York, NY, USA, 2010. ACM.

[56] Leah Larkey and W. Bruce Croft. Combining classifiers in text categorization. pages 289–297. ACM Press, 1996.

[57] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 37–50, New York, NY, USA, 1992. ACM.

[58] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. pages 80–86, 1998.

[59] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification, 1998.

[60] Dunja Mladenic and Janez Brank. Feature selection using linear classifier weights: interaction with classification models. In *In Proceedings of the 27th Annual International ACM SIGIR Conference (SIGIR2004*, pages 234–241. ACM Press, 2004.

[61] Paul Ogilvie, , Paul Ogilvie, and Jamie Callan. Experiments using the lemur toolkit. In *In Proceedings of the Tenth Text Retrieval Conference (TREC-10*, pages 103–108, 2002.

[62] Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 264–271, New York, NY, USA, 2000. ACM.

[63] Masayuki Okabe and Seiji Yamada. Semisupervised query expansion with minimal feedback. *IEEE Transactions on Knowledge and Data Engineering*, 19:1585–1589, 2007.

[64] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[65] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[66] Mehran Sahami. Learning limited dependence bayesian classifiers. In *In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338. AAAI Press, 1996.

[67] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail, 1998.

[68] Thiago Salles, Leonardo Rocha, Gisele L. Pappa, Fernando Mourão, Wagner Meira, Jr., and Marcos Goncalves. Temporally-aware algorithms for document classification. In *Proceedings of the 33rd international ACM SIGIR conference*

on Research and development in information retrieval, SIGIR '10, pages 307–314, New York, NY, USA, 2010. ACM.

[69] G. Salton and M. J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[70] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, March 2002.

[71] Vikas Sindhwani and S. Sathiya Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 477–484, New York, NY, USA, 2006. ACM.

[72] Noam Slonim and Naftali Tishby. The power of word clusters for text classification. In *In 23rd European Colloquium on Information Retrieval Research*, 2001.

[73] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. Indri: a language-model based search engine for complex queries. Technical report, in Proceedings of the International Conference on Intelligent Analysis, 2005.

[74] Jian-Tao Sun, Zheng Chen, Hua-Jun Zeng, Yu-Chang Lu, Chun-Yi Shi, and Wei-Ying Ma. Supervised latent semantic indexing for document categorization. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 535 – 538, nov. 2004.

[75] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[76] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting, 1995.

[77] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20:1–20:36, September 2012.

[78] Shuang-Hong Yang and Hongyuan Zha. Language pyramid and multi-scale text analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 639–648, New York, NY, USA, 2010. ACM.

[79] Hongyuan Zha and Horst D. Simon. On updating problems in latent semantic indexing. *SIAM J. Sci. Comput.*, 21(2):782–791, September 1999.

[80] Huaiyu Zhu. On information and sufficiency, 1997.