Cent. Eur. J. Comp. Sci. • 1-18
Author version

Central European Journal of Computer Science

Word Sense Disambiguation to Improve Precision for Ambiguous Queries

Research Article

Adrian-Gabriel Chifu[1]*, Radu-Tudor Ionescu[2]†

1 IRIT UMR5505, CNRS, Université de Toulouse, Université Paul Sabatier,
118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9, France

2 University of Bucharest, Faculty of Mathematics and Computer Science,
Academiei 14, RO-010014, Bucharest, Romania

Abstract: Success in Information Retrieval (IR) depends on many variables. Several interdisciplinary approaches try to improve the quality of the results obtained by an IR system. In this paper we propose a new way of using word sense disambiguation (WSD) in IR. The method we develop is based on Naïve Bayes classification and can be used both as a filtering and as a re-ranking technique. We show on the TREC ad-hoc collection that WSD is useful in the case of queries which are difficult due to sense ambiguity. Our interest regards improving the precision after 5, 10 and 30 retrieved documents (P@5, P@10, P@30), respectively, for such lowest precision queries.

Keywords: Information Retrieval • Word Sense Disambiguation • Naïve Bayes classification • difficult queries • ambiguous queries • document clustering • fusion functions

1. Introduction

The concept of "difficult query", the characteristics of such queries, as well as the ways to identify them, have been studied in the information retrieval (IR) literature for a long time [4, 9]. Some authors [14, 18] find promising links between various linguistic characteristics and query difficulty. They show that linguistic features of a query are good indicators to predict a systems' failure to answer it. In [17] it is shown that the most significant such features are "syntactic complexity (in terms of distance between syntactically linked words) and word polysemy (in terms of number of semantic classes a given word belongs to)".

The present paper concentrates on queries that are difficult due to polysemy, which leads to sense ambiguity, and proposes using word sense disambiguation (WSD) techniques in IR. Our aim is to analyze semantically

* *E-mail: adrian.chifu@irit.fr*
† *E-mail: raducu.ionescu@gmail.com*

ambiguous queries that lead to system failure. Polysemy of a query word will be denoted by its occurrence in multiple WordNet (WN) synsets [6], [16].

As it is well known, IR is based on query terms and document term matching, without considering the meaning of terms. Our approach takes into consideration the semantics carried out both by the queries and by the documents. Terms will not be considered as independent, but rather as closely related. The main assumption of this study is that context could improve the performance of IR systems.

Whether or not WSD can improve the results of IR has been a longtime source of debate among specialists. Over time, several authors have concluded that WSD does not allow significant retrieval performance improvement [8, 21]. Various studies [13, 21, 23] have argued that the main problem in improving retrieval performance when using WSD lies in the inefficiency of the existing disambiguation algorithms, a problem which increases in the case of short queries (one or two words).

The choice of the WSD algorithm to be used in this type of framework is not an easy one. Let us begin by noting that several authors investigating the problem have concentrated on a straightforward disambiguation of the ambiguous words of a query and/or document, namely on the actual senses of these words. The approach of [8], for instance, which is extensively cited in the literature, views the disambiguation process as consisting of "annotating documents and queries by adding sense information to all content words". Corresponding to each occurrence of a word, these authors provide a list of senses together with their respective scores. The senses are represented by WN synsets, with only the synset that has the best score being retained. This type of approach to WSD is knowledge-based and refers to the actual meaning of an ambiguous word.

Unlike these authors, we are here suggesting and investigating the usage of an unsupervised WSD technique. We believe that IR is an application for which this type of analysis is useful. Our approach will not be concerned with performing a straightforward WSD, but rather with discriminating among the meanings of a polysemous word by identifying clusters of similar contexts, where each cluster shows that word being used in a particular meaning. This type of approach (sense discrimination) is quite distinct from the traditional task of WSD, which classifies words relative to existing senses. To solve the proposed IR problem we are therefore relying on clustering the retrieved documents.

From the wide range of unsupervised learning techniques that could be applied to our problem, we have chosen to use a parametric model to assign a sense group to an ambiguous occurrence of a so-called target word. In each case we shall assign the most probable group given the context as defined by the Naïve Bayes model, where the parameter estimates are formulated via unsupervised techniques. WN-based feature selection, which has provided good disambiguation results in the case of all parts of speech [10, 11], will be used.

The idea of using a Bayesian classifier in the context of WSD is that it looks at the words around an ambiguous word within the so-called context window. Each content word contributes with useful information concerning which sense of the ambiguous word is more likely to be used with it. The classifier does no feature selection. Instead it combines the evidence coming from all features [15], namely from all content words occurring in the

context window of the target. A fact that generates a so-called "bag of words model", based on the Naïve Bayes assumption that all these content words are conditionally independent. The assumption is clearly not true in the case of natural language, but has provided very good practical disambiguation results. As commented in [15], there is a surprisingly large number of cases in which the Naïve Bayes assumption does well, "partly because the decisions made can still be optimal even if the probability estimates are inaccurate due to feature dependence". Using this classifier as clustering technique, our aim is to improve precision for top-5, top-10 and top-30 (namely the first 5, 10 and 30 retrieved documents, respectively).

The paper is organized as follows: in section 2, we present the mathematical model to disambiguate terms and we discuss its implementation. The described technique is used to cluster the documents initially retrieved by the baseline IR system, with the number of clusters being given by the number of senses of the ambiguous terms. The query itself is treated as a document and is included in one of the obtained clusters. Each query term is disambiguated by retaining only the corresponding cluster of documents which contains the query. Section 3 uses these clusters to re-rank the initially retrieved documents in order to improve precision. Section 4 presents the evaluation framework. The obtained results are presented in section 5. Section 6 concludes this paper.

2. The WSD method and its implementation

Our purpose is to study if and how WSD can improve the performance of an IR system, relative to queries which contain ambiguous words. The approach we promote aims at re-ordering an initially retrieved document list by boosting documents that are semantically similar to queries. To start with, corresponding to each query, we retrieve a set of documents by means of a search engine and we order these documents according to their scores. Our objective then becomes to pinpoint the documents which are more relevant to the information need and to enable them to obtain a better position at the top of the document list.

The first step of the method is based on clustering the retrieved set of documents with respect to the senses of each ambiguous word. More precisely, for each query and for each ambiguous term occurring in that query, we cluster the retrieved documents into a number of clusters equal to the number of senses an ambiguous term has, according to WN, which will be used as sense inventory. In the second step of the method, we disambiguate each term by keeping only the cluster of documents which also contains the query.

2.1. Unsupervised WSD with an underlying Naïve Bayes model

The algorithm for WSD that is used here exemplifies an important theoretical approach in statistical language processing: Bayesian classification [7].

In order to formalize the Bayesian model in the context of WSD, we shall follow [10] and we shall present the probability structure of the corpus $\mathcal{C}$. The following notations will be used: w is the word to be disambiguated (target word); $s_1, ..., s_K$ are possible senses for w;; the $c_1, ..., c_I$ are contexts of w in a corpus $\mathcal{C}$; and the $v_1, ..., v_J$

are words used as contextual features for the disambiguation of w.

2.1.1. The probability model of the corpus, the Bayes classifier and parameter estimation

Let us note that the contextual features could be some attributes (morphological, syntactical, etc.), or they could be actual "neighboring" content words of the target word. The contextual features occur in a fixed position near w, in a window of fixed length, centered or not on w. In what follows, a window of size n will denote taking into consideration n content words to the left and n content words to the right of the target word, whenever possible. The total number of words taken into consideration for disambiguation will therefore be $2n+1$. When not enough features are available, the entire sentence in which the target word occurs will represent the window of context. The probability structure of the corpus is based the assumption that *the contexts* $\{c_i, i = 1, ..., I\}$ *in the corpus* $\mathcal{C}$ *are independent.* Hence, the likelihood of $\mathcal{C}$ is given by the product

$$P(\mathcal{C}) = \prod_{i=1}^{I} P(c_i)$$

This assumption is quite natural, as the contexts are not connected because they occur at significant lags in $\mathcal{C}$. Considering the possible senses of each context, one gets

$$P(\mathcal{C}) = \prod_{i=1}^{I} \sum_{k=1}^{K} P(s_k) \cdot P(c_i \mid s_k)$$

A model with independent features (usually known as the Naïve Bayes model) assumes that the contextual features are conditionally independent. That is,

$$P(c_i \mid s_k) = \prod_{v_j \ in \ c_i} P(v_j \mid s_k) = \prod_{j=1}^{J} (P(v_j \mid s_k))^{|v_j \ in \ c_i|},$$

where $|v_j \ in \ c_i|$ denote the number of occurrences of feature v_j in context c_i. The likelihood of the corpus $\mathcal{C}$ is then

$$P(\mathcal{C}) = \prod_{i=1}^{I} \sum_{k=1}^{K} P(s_k) \prod_{j=1}^{J} (P(v_j \mid s_k))^{|v_j \ in \ c_i|}$$

The parameters of the probability model with independent features are

$$\{P(s_k), k = 1, ..., K \ \ and \ \ P(v_j \mid s_k), j = 1, ..., J, k = 1, ..., K\}$$

Notation:

- $P(s_k) = \alpha_k$, $k = 1, ..., K$, $\alpha_k \geq 0$ for all k, $\sum_{k=1}^{K} \alpha_k = 1$
- $P(v_j \mid s_k) = \theta_{kj}$, $k = 1, ..., K$, $j = 1, ..., J$, $\theta_{kj} \geq 0$ for all k and j, $\sum_{j=1}^{J} \theta_{kj} = 1$ for all $k = 1, ..., K$

With this notation, the likelihood of the corpus $\mathcal{C}$ can be written as

$$P(\mathcal{C}) = \prod_{i=1}^{I} \sum_{k=1}^{K} \alpha_k \prod_{j=1}^{J} (\theta_{kj})^{|v_j \ in \ c_i|}$$

The well known Bayes classifier involves a posteriori probabilities of the senses, calculated by the Bayes formula for a specified context c,

$$P(s_k \mid c) = \frac{P(s_k) \cdot P(c \mid s_k)}{\sum_{k=1}^{K} P(s_k) \cdot P(c \mid s_k)} = \frac{P(s_k) \cdot P(c \mid s_k)}{P(c)},$$

with the denominator independent of senses.

The Bayes classifier chooses the sense s' for which the a posteriori probability is maximal (sometimes called the Maximum A Posteriori classifier)

$$s' = \arg\max_{k=1,\ldots,K} P(s_k \mid c)$$

Taking into account the previous Bayes formula, one can define the Bayes classifier by the equivalent formula

$$s' = \arg\max_{k=1,\ldots,K} (\log P(s_k) + \log P(c \mid s_k))$$

Of course, when implementing a Bayes classifier, one has to estimate the parameters first.

Parameter estimation is performed by the Maximum Likelihood Method, for the available corpus $\mathcal{C}$. That is, one has to solve the optimization problem

$$\max (\log P(\mathcal{C}) \mid \{P(s_k), k = 1, ..., K \ \ and \ \ P(v_j \mid s_k), j = 1, ..., J, k = 1, ..., K\})$$

For the Naïve Bayes model, the problem can be written as

$$\max \left(\sum_{i=1}^{I} \log \left(\sum_{k=1}^{K} \alpha_k \prod_{j=1}^{J} (\theta_{kj})^{|v_j \ in \ c_i|} \right) \right) \tag{1}$$

with the constraints

$$\sum_{k=1}^{K} \alpha_k = 1$$
$$\sum_{j=1}^{J} \theta_{kj} = 1 \ \ for \ \ all \ \ k = 1, ..., K$$

For unsupervised disambiguation, where no annotated training corpus is available, the maximum likelihood estimates of the parameters are usually [10], [11], [12], [19] constructed by means of the Expectation - Maximization (EM) algorithm [5]. The EM algorithm is known as a very successful iterative method, very well fitted for models

with missing data (which, in our case, are the senses of the ambiguous words). It has been used by us as well for parameter estimation, closely following [12].

If, for simplicity, we denote the vector of parameters[1] by

$$\psi = (\alpha_1, ..., \alpha_K, \theta_{11}, ..., \theta_{KJ})$$

then it is well known that the EM iterations $\left(\psi^{(r)}\right)_r$ converge to the Maximum Likelihood Estimate

$$\widehat{\psi} = \left(\widehat{\alpha}_1, ..., \widehat{\alpha}_K, \widehat{\theta}_{11}, ..., \widehat{\theta}_{KJ}\right).$$

Once the parameters of the model have been estimated[2], we can disambiguate contexts of w by computing the probability of each of the senses based on features v_j occurring in the context c. Making the Naïve Bayes assumption and using the Bayes decision rule, we can decide s' if

$$s' = \arg\max_{k=1,...,K} \left(\log \widehat{\alpha}_k + \sum_{j=1}^{J} |v_j \ in \ c| \cdot \log \widehat{\theta}_{kj} \right)$$

When the Naïve Bayes model is applied to supervised disambiguation, the actual words occurring in the context window are usually used as features. This type of framework generates a great number of features and, implicitly, a great number of parameters. As noted in [10], this can dramatically decrease the model performance since the available data is usually insufficient for the estimation of the great number of resulting parameters. A situation that becomes even more drastic in the case of unsupervised disambiguation, where parameters must be estimated in the presence of missing data (the sense labels). In order to overcome this problem, the various existing unsupervised approaches to WSD implicitly or explicitly perform a feature selection. Of the possible ways of performing feature selection the present study implements WN-based feature selection as described in [10]. This approach to WSD places the disambiguation process at the border between unsupervised and knowledge-based techniques. It is based on a set of features formed by the actual words occurring near the target word (within the context window) and reduces the size of this feature set by performing knowledge-based feature selection that relies entirely on WN. The WN semantic network provides the words considered relevant for the set of senses taken into consideration corresponding to the target word. In our experiments, WordNet 3.0 has been used.

First of all, words occurring in the same WN synsets as the target word (WN synonyms) have been chosen, corresponding to all senses of the target. Additionally, the words occurring in synsets related (through explicit relations provided in WN) to those containing the target word have also been considered as part of the vocabulary used for disambiguation. Synsets and relations were restricted to those associated with the part of speech of the

[1] *Let us notice that the number of independent components (parameters) is* $(K-1)+(KJ-K) = KJ-1$.
[2] *For a detailed presentation of the involved computation, see [12].*

target word [10]. The content words of the glosses of all types of synsets participating in the disambiguation process, using the corresponding example strings as well, have equally been taken into consideration. The latter choice has been made since previous studies [2], performed for knowledge-based disambiguation, have come to the conclusion that the "example relation" (which simply returns the example string associated with the input synset) seems to provide useful information in the case of all parts of speech.

2.2. Clustering the retrieved documents

Before we can apply the described WSD technique to the queries, we need to process the data within a preprocessing step. The preprocessing step identifies the polysemous words of a query (as a result of their occurrence in multiple WN synsets) and builds the feature sets for them (using the same WN semantic network). Prior to building the feature sets, we also need to determine the part-of-speech of the identified polysemous words, since features depend on the part-of-speech of the target word.

Corresponding to each query we have considered the first 1000 documents retrieved by the IR system. For each document we also know the score which indicates how similar that document is to a specific query. A "special document" representing the query itself is added to this set of documents. During the data preprocessing stage the feature set of each document is determined by creating an incidence matrix with rows representing the documents and columns representing the features. Each element of this matrix is either 1 or 0, if the feature indicated by the column index is either present or not present in the document indicated by the row index, respectively. The number of WN senses and the incidence matrix obtained after data preprocessing will be used as input for the WSD algorithm. Thus, the WSD process is performed on 1001 documents (with one document representing the query).

Let us now describe the entire WSD process corresponding to a single polysemous target word. The first step is to build the feature set using the WN lexical knowledge base, starting from the synsets of the target. Besides the WN synonyms and the content words of the glosses (examples included) that are always part of the feature set, we have equally included into the disambiguation vocabulary words indicated by specific WN semantic relations that depend on the part-of-speech of the target. For nouns we have considered hyponyms, meronyms, hypernyms and holonyms; for adjectives we have considered similar synsets, antonyms, attributes, pertainyms and related (see also) synsets; for verbs we have considered troponyms, hypernyms, entailments and outcomes; for adverbs we have considered antonyms, pertainyms and topics.

Let us note that we have taken into account two ways of choosing the semantic relation set in the case of nouns. If, corresponding to the target noun, over 200 features are obtained when considering all four mentioned relations, then only the hyponymy and meronymy relations were used for building the feature set. Previous studies performed for knowledge-based disambiguation [2] show that using hyponymy and meronymy improves the WSD accuracy by 5% compared to using all four relations. By using only the hyponymy and meronymy relations, the feature set will be smaller, thus enabling more accurate estimation by the EM algorithm. However, if the number

of features generated by all four relations is lower than 200, then the obtained feature set is kept unchanged in our experiments. In this case and from the point of view of the relations used for feature selection, the WSD accuracy may drop, but, in practice, having more features increases the probability of feature occurrence in a specific context. The accuracy of the WSD algorithm increases as the number of contexts that contain features increases. Keeping the feature set based on all four relations for nouns will ensure a greater number of contexts with features, namely greater coverage. Even if the accuracy of the unsupervised WSD algorithm drops to around 60-70%, the algorithm will overcome the problem of the existing high number of contexts without features. This problem is caused by WN feature selection that is meant to reduce the size of the feature set. Reduction of the feature set size should, however, not be performed without taking into account corpus coverage.

The set of words generated by WN feature selection needs to be processed in order to obtain the final set of features. This processing ensures a closed and uniform feature set. The first processing step is to eliminate the stopwords. The remaining words are stemmed using the Porter stemmer algorithm [20][1]. This algorithm removes the commoner morphological and inflexional endings from words in English. Next, we eliminate the stem of the target word. The remaining stems, alphabetically ordered, represent the final set of features that we use in the WSD process. This is the so-called "disambiguation vocabulary" [10].

The second step is to build the incidence matrix that indicates what features occur in each document. First, we eliminate the stopwords from each document, then we determine the position of the target word within each document. If a certain document contains the target several times, we shall consider only its first occurrence. In our experiments we have used a context window of size 25, as suggested for obtaining the best possible disambiguation accuracy in [10]. A context word is included in the context window only if it occurs in the WN network, a fact which ensures that the respective word can also occur in the feature set (that was generated using words coming solely from WN). For each document we verify if the words in the context window are also found in the feature set. The features that occur in the context window are stored in the row of the matrix that corresponds to the analyzed document.

The final result of the WSD process for a specific polysemous word consists in determining the document clusters relative to that word. Each obtained cluster corresponds to a certain sense of the polysemous word, with one of the clusters containing the query itself. The WSD process associates a sense to a polysemous word of a query by placing the query in a cluster with similar documents. Note that two documents are similar, from the WSD point of view, if the polysemous word has the same sense in both documents. We should point out that our unsupervised WSD method does not give the actual word sense because we do not know which cluster refers to a specific sense. However, it is not necessary to pair clusters with senses, as document clusters are enough for explicit automatic disambiguation in IR.

[1] *Stemming is the process that reduces a word to its root form.*

3. Merging the clustering results with the results obtained by the baseline IR system

Our approach aims at improving the top precision measures (P@5, P@10 and P@30) as a result of performing WSD. The first step of the discussed method results in a set of documents extracted by the search engine corresponding to each query and in the clusters of documents obtained in Section 2.2. The cluster which also includes the query contains documents having the same sense of the polysemous term as the query. Therefore, disambiguating polysemous terms results in retaining only those documents occurring in the same cluster as the query.

A query can contain several ambiguous terms. In this case, as many clusters of documents as the number of ambiguous words in the query are retained. In order to form a unique list of documents, we fuse these sets of documents, for which we consider as document scores the values obtained by the search engine. Various fusion functions that can be used for this purpose have been defined in the literature [22].

At this stage, we may already state that our method can be used as a filtering technique. However, one can go further in order to obtain a re-ranking method as well. In doing so, we shall modify the order of the retrieved documents, by boosting the documents that are semantically similar to the query, as will be described in what follows.

Indeed, our main purpose for using a WSD technique in IR was to find the most probable relevant documents and to assign them a higher rank in the initial document list. The way to reach this latter goal is to improve the scores of those (relevant) documents by using the WSD results. To do this, we fuse the initial set of documents with the ones obtained as a result of clustering. According to the discussed method, the documents obtained by the search engine and the set of documents obtained after clustering have different importance in the final results. We therefore weigh the fusion function with a parameter.

Finally, we split the set of queries into quartiles using the baseline precision after the first 5 documents (P@5) as criteria. We apply our method only corresponding to the queries with the lowest performance (those from the first quartile), while the other queries remain untouched. This results in a re-ranking technique and in a method to treat queries with the lowest precision as well.

4. Data and baseline

To test the proposed method we have used the TREC 7 and TREC 8 ad-hoc datasets. As a baseline we have used the performance of the Terrier search engine with customized parameters. In the following subsections we shall briefly describe the TREC dataset as well as Terrier with its parameters.

4.1. The TREC 7 and TREC 8 ad-hoc collections

TREC (Text REtrieval Conference) is an annual workshop hosted by the US National Institute of Standard and Technology which provides the infrastructure necessary for large-scale evaluation of text retrieval methods[2].
The ad-hoc tasks investigate the performance of systems that search a static set of documents using new questions (called topics). Not only should such systems answer well to "easy" queries, but also to "difficult" ones (a query is considered as difficult when most of the systems have failed on it). The competition provides approximately 2 gigabytes worth of documents and a set of natural language topic statements. The documents represent articles from newspapers like the *Financial Times*, *Federal Register*, the *Foreign Broadcast Information Service* and the *Los Angeles Times*. A number of 50 topics have been selected by us for this task (from 351 to 400 for TREC 7 and from 401 to 450 for TREC 8, respectively).

4.1.1. TREC topics

TREC distinguishes between a statement of information need (the topic) and the data structure that is actually given to a retrieval system (the query). The TREC test collections provide topics. What is now considered the "standard" format of a TREC topic statement comprises a topic id, a title, a description, and a narrative. The title contains two or three words that represent the key words a user could have used to query the IR system. The description contains one or two sentences that describe the topic area. The narrative part gives a concise description of what makes a document relevant [24]. Both the descriptive and the narrative parts can offer clues about the word senses used in the title part. In the described WSD process all three parts will be used to form the context window for the ambiguous words in the title.

4.1.2. TREC evaluation

In TREC, ad-hoc tasks are evaluated using the `trec_eval` package. This package reports various performance measures, including recall and precision at various cut-off levels plus single valued summary measures that are derived from recall and precision. Precision is the proportion of retrieved documents that are relevant (number-retrieved-and-relevant/number-retrieved), while recall is the proportion of relevant documents that are retrieved (number-retrieved-and-relevant/number-relevant). A cut-off level is a rank that defines the retrieved set; for example, a cut-off level of ten defines the retrieved set as the top ten documents in the ranked list. Our study uses three cut-off levels: P@5, P@10 and P@30. Despite these measures are strongly correlated [1], they give details on high precision results. The `trec_eval` program reports the scores as averages over the set of topics, where each topic is equally weighted.
The present study is based on runs constructed by Terrier. It uses the first 1000 documents retrieved for each topic.

[2] *http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=10667*

4.2. Terrier

Terrier is an open source search engine that implements state-of-the-art indexing and retrieval functionalities. Terrier is developed at the School of Computing Science, University of Glasgow[3].

The configuration of Terrier's parameters that we have used is the following: we have demanded a two step indexation (both direct and inverted index), the indexation by block was activated and we have used BB2c1.0 as a normalization parameter. As a query expansion model our choice was the KLbfree model. The configuration took 3 documents to be used for the query expansion. A term has to appear in two documents in order to be considered relevant. Finally, the number of terms to be added to the query for the process of query expansion was set between 10 and 20. We have chosen this configuration since it was the best -in terms of MAP- that we obtained for the TREC7 ad-hoc collection. On the other hand, for TREC 8 the parameter configuration stays in place, except for the query expansion model which is changed with the DFree model.

5. Evaluation

We have tested our method both as a filtering technique and as a re-ranking one. We discuss here the choices we have made and the obtained results. We compare our performances with the considered baseline.

5.1. The filtering method for all the ambiguous queries

Our WSD analysis has shown that the queries (topic title) taken into account for our study may contain from zero up to three ambiguous words. In the case of the TREC 7 data, the analysis results for the considered topics show 15 queries with no polysemous words, 22 queries with only one polysemous word, 10 queries with two polysemous words and 3 queries with three polysemous words. Corresponding to the TREC 8 data, the analysis results identify 15 queries with no polysemous words, 22 queries with only one polysemous word, 7 queries with two polysemous words and 6 queries with three polysemous words.

The average number of senses for the polysemous words of the 35 TREC 7 ambiguous queries is 3.47 and the average number of features is 128.5. There are three adjectives (human, commercial, organic) and two verbs (teaching, dismantling), the rest of the polysemous words being nouns. In the case of the TREC 8 data, the average number of senses is 4.03 and the average number of features is 138.07. There are four adjectives, five verbs and forty five nouns. In the preprocessing stage we did not identify any adverbs, both corresponding to the TREC 7 and to the TREC 8 data, this fact being consistent with other studies in natural language processing finding that adverbs, in general, are less ambiguous and more rare in natural language than other parts of speech. The English nouns, however, have a high degree of polysemy; even two proper names that are polysemous have

[3] *http://www.terrier.org*

been detected in both collections: "El Nino" and "Amazon" in TREC 7 and "Cuba" and "Triangle" (from the Golden Triangle) in TREC 8, respectively.

We have considered as scores for the list of documents forming the clusters the initial scores obtained by Terrier. Regarding the fusion function, we have evaluated CombSum and CombMNZ [22].

The CombMNZ function computes the final scores for documents as follows:

$$S_f = (S_1 + S_2)/c \ ,$$

where S_f represents the final score for each document, S_1 represents the score of a document from the Terrier set of documents and S_2 represents the score of the same document in the clustering set of documents (if the document does not exist in this set, $S_2 = 0$), and c represents the number of nonzero scores for each document ($c = 1$ if the document occurs only in one of the two sets of documents and $c = 2$ if the document occurs in both of these sets).

The CombSum function has the formula

$$S_f = (S_1 + S_2) \ ,$$

which means that it does not normalize the score by the number of occurrences as CombMNZ does, so it boosts the scores of those documents which occur in both clusters.

We have constructed TREC runs for our filtering method using both of these fusion functions. We present the obtained results for the TREC 7 data in Table 1. The same results corresponding to the TREC 8 data are presented in Table 2.

Table 1. **Baseline, CombSum and ComMNZ results for all the ambiguous queries (TREC 7)**

Precision	Baseline	CombSum	CombMNZ
P@5	0.6343	0.5257	0.5429
P@10	0.5600	0.4629	0.4914
P@30	0.4095	0.3571	0.3714

Table 2. **Baseline, CombSum and ComMNZ results for all the ambiguous queries (TREC 8)**

Precision	Baseline	CombSum	CombMNZ
P@5	0.4960	0.4171	0.4686
P@10	0.4660	0.3514	0.3829
P@30	0.3707	0.2476	0.2752

The CombMNZ function behaves constantly better than the CombSum function, but the baseline is not even reached (both in the case of the TREC 7 data and in that of the TREC 8 data) because of the possible loss of relevant documents after filtering. Therefore, we evaluate our method as a re-ranking method.

5.2. The re-ranking method

5.2.1. All the ambiguous queries

The re-ranking method is based on an additionnal fusion step: the initial set of documents, retrieved by Terrier, is fused with the list of documents obtained by clustering. We test the most favorable contribution of the two sets of documents to the final results by parameterizing the fusion function. The parameterized fusion function has the following structure:

$$S_f = S_1 + \alpha S_2$$

where S_f represents the final score for each document, S_1 represents the score of a document from the Terrier set of documents, S_2 represents the score of the same document in the clustering set of documents (if it exists in this set) and $\alpha \in [0, 1]$ represents the weight of the clustering method for the final results. Let us mention that we have started with $\alpha = 0$ and that we have increased this parameter by 0.01 at each trial. To evaluate our decision we have constructed the corresponding TREC runs, and we have determined the overall precisions for each alpha parameter value.

Figures 1, 2 and 3 show that the higher the alpha parameter value is, the lower is the performance. There are some improvements for alpha between 0.1 and 0.2, but the difference between the results and the baseline is usually less than 0.01. This conclusion holds for both the TREC 7 and the TREC 8 test data.

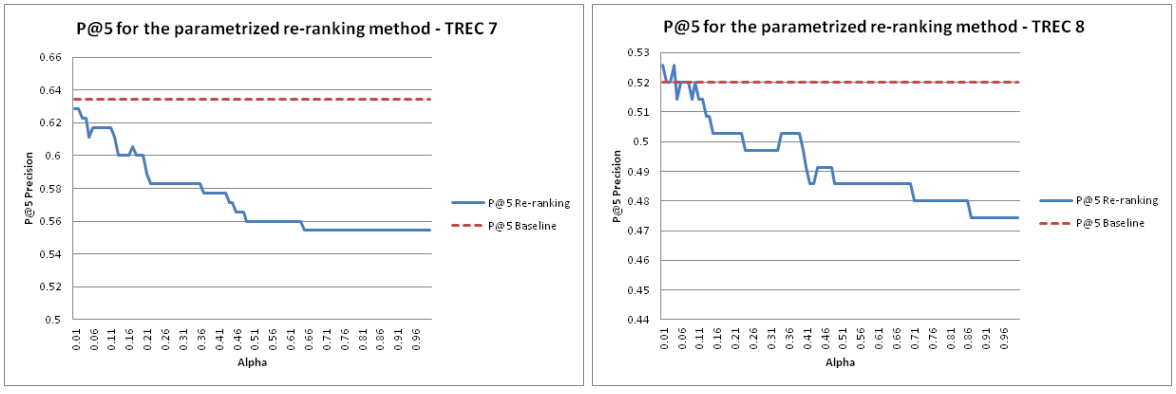

Figure 1. Precision after the first 5 retrieved documents, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

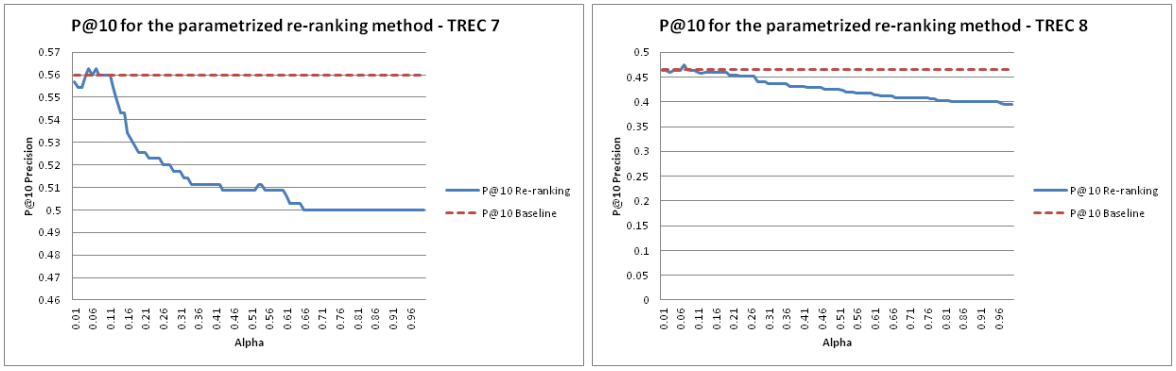

Figure 2. Precision after the first 10 retrieved documents, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

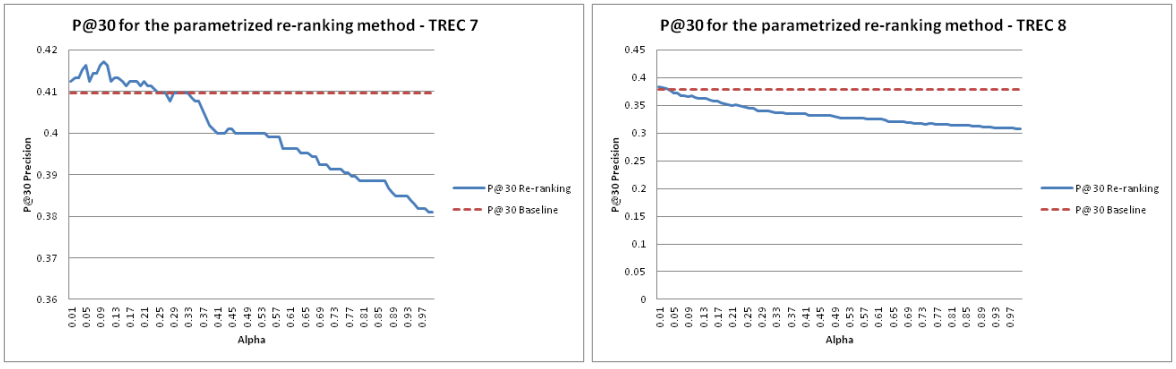

Figure 3. Precision after the first 30 retrieved documents, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

5.2.2. Lowest precision queries

Rather than evaluating the result on all the ambiguous queries, we consider poorly performing topics only. Previous work have shown that clustering topics according to their difficulty in terms of precision provides interesting insight [3].

With respect to quartiles, we have considered the TREC 7 35 ambiguous queries and the TREC 8 35 ambiguous queries, respectively and have grouped them by their overall precision for the top 5 documents obtained with Terrier. The first quartile, the one with the lowest precision, contains 9 queries (TREC 7) and 11 queries (TREC 8), respectively.

We present the corresponding results of the parameterized fusion function in Figures 4, 5 and 6.

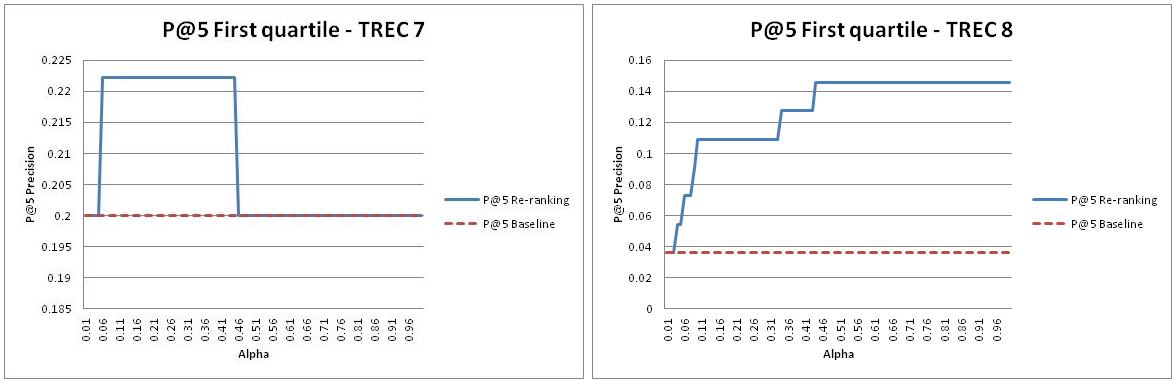

Figure 4. Precision after the first 5 retrieved documents, obtained for the lowest precision queries, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

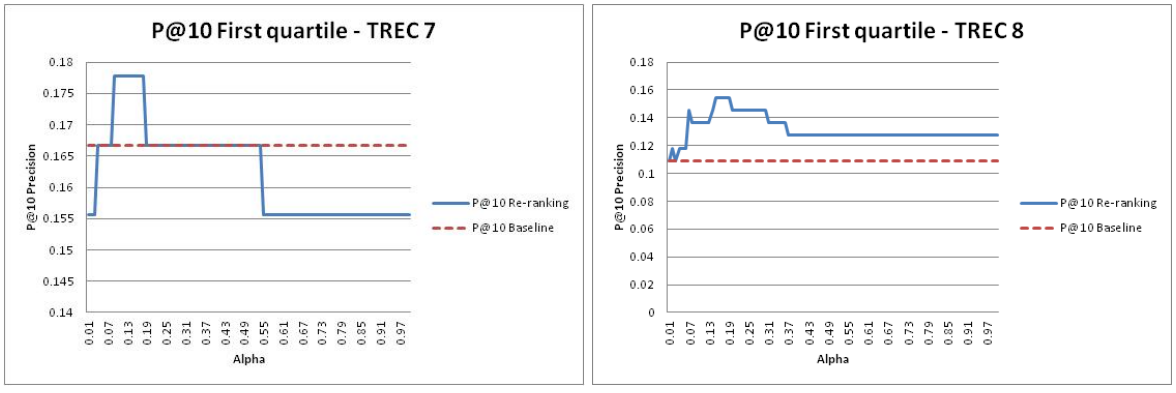

Figure 5. Precision after the first 10 retrieved documents, obtained for the lowest precision queries, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

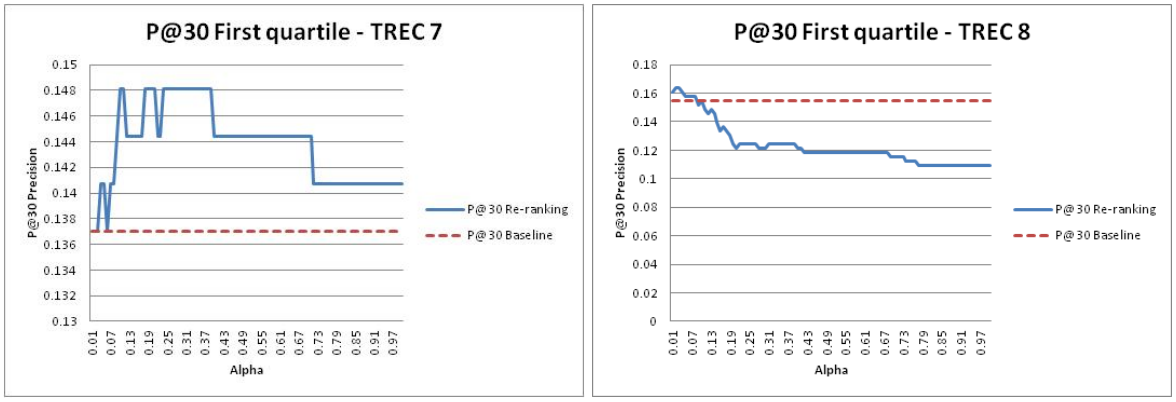

Figure 6. Precision after the first 30 retrieved documents, obtained for the lowest precision queries, compared to the baseline, for the parametrized method (TREC 7 and TREC 8)

The best results were obtained with the alpha parameter having values between 0.1 and 0.2 in the case of both datasets. Since alpha is the parameter assigning a weight to the clustering method for the final results, one can notice that it is best for the cluster documents to participate with 1/5 of their scores in the final document score. Regarding the TREC 8 results we can notice, at P@5, a significant improvement of precision, compared to the baseline, for all the values of alpha greater than 0.03. In this case, the highest difference between our results and the baseline is 0.1091.

The baseline precisions and the best results obtained for the lowest precision queries are shown in Table 3 (TREC 7) and in Table 4 (TREC 8), respectively.

Table 3. **Baseline and best results for the lowest precision queries (TREC 7)**

Precision	Baseline	Best results
P@5	0.2000	0.2222
P@10	0.1667	0.1778
P@30	0.1370	0.1481

Table 4. **Baseline and best results for the lowest precision queries (TREC 8)**

Precision	Baseline	Best results
P@5	0.0364	0.1455
P@10	0.1091	0.1545
P@30	0.1545	0.1636

6. Conclusions and future work

We have obtained both a filtering method and a re-ranking method for IR, with significant improvement of high precision for ambiguous queries. We consider this a most important aspect, keeping in mind the fact that IR systems are predisposed to failure in the case of this particular type of queries [17, 18].

Despite the existence of several "skeptical studies" [8, 21], we believe that WSD still holds a promise for IR. In future studies we shall extend our research to other datasets and we shall try to further improve IR results (with special reference to ambiguous queries) by changing the clustering technique, when moving from classical Naïve Bayes to other, state-of-the-art clustering methods.

Acknowledgments

The authors express their deepest gratitude to Professor Josiane Mothe of Université de Toulouse for constant guidance during all stages of this research.

References

[1] Baccini A., Déjean S., Lafage L., Mothe J., How many performance measures to evaluate Information Retrieval Systems?, Knowledge and Information Systems, 2012, 30(3), 693–713

[2] Banerjee S., Pedersen T., Extended Gloss Overlaps as a Measure of Semantic Relatedness, In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, (Stockholm Sweden), 2003, 805–810

[3] Bigot A., Chrisment C., Dkaki T., Hubert G., Mothe J., Fusing different information retrieval systems according to query-topics: a study based on correlation in information retrieval systems and TREC topics, Inf. Retr., 2011, 14(6), 314–648

[4] Cronen-Townsend S., Zhou Y., Croft W.B., Predicting Query Performance, In: Proceedings of the 25th annual international ACM-SIGIR conference on research and development in information retrieval, (New-York USA), ACM Press, 2002, 299–306

[5] Dempster A., Laird N., Rubin D., Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B, 1977, 39(1), 1–38

[6] Fellbaum C., (Eds.), WordNet: an Electronic Lexical Database, The MIT Press, Cambridge, MA, 1998

[7] Gale W., Church K., Yarowsky D., A method for disambiguating word senses in a large corpus, Computers and the Humanities, 1992,26(5-6), 415–439

[8] Guyot J., Falquet G., Radhouani S., Benzineb K., Analysis of word sense disambiguation-based information retrieval, In: Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access, (Aarhus Denmark), 2008, 146–154

[9] Harman D., Buckley C., Overview of the Reliable Information Access Workshop, Inf. Retr., 2009, 12(6), 615–641

[10] Hristea F., Popescu M., Dumitrescu M., Performing word sense disambiguation at the border between unsupervised and knowledge-based techniques, Artificial Intelligence Review, 2008, 30(1-4), 67–86

[11] Hristea F., Recent Advances Concerning the Usage of the Naïve Bayes Model in Unsupervised Word Sense Disambiguation, International Review on Computers and Software, 2009, 4(1), 58–67

[12] Hristea F., Popescu, M., Adjective Sense Disambiguation at the Border Between Unsupervised and Knowledge-Based Techniques, Fundamenta Informaticae, 2009, 91, 547–562

[13] Krovetz R., Croft W. B., Lexical ambiguity and information retrieval, ACM TOIS, 1992, 10(2), 115–141

[14] Mandl T., Womser-Hacker C., Linguistic and Statistical Analysis of the CLEF Topics, CLEF Workshop, (Rome Italy), Springer, 2002, 505–511

[15] Manning C., Schütze H., Foundations of Statistical Natural Language Processing, Cambridge, MA: The MIT Press, 2003

[16] Miller G. A., Nouns in WordNet: a lexical inheritance system, International Journal of Lexicography, 1990, 3(4), 245–264

[17] Mothe J., Tanguy L., Linguistic features to predict query difficulty - a case study on previous TREC campaigns, In: SIGIR, Predicting query difficulty - methods and applications workshop, (Salvador Bahia Brazil), 2005, 7–10

[18] Mothe J., Tanguy L., Linguistic Analysis of Users' Queries: towards an adaptive Information Retrieval System, In: International Conference on Signal-image technology & Internet-Based System, (Shanghai China), 2007, 77–84

[19] Pedersen T., Bruce R., Knowledge Lean Word-Sense Disambiguation, In: Proceedings of the 15th National Conference on Artificial Intelligence, AAAI Press, 1998, 800–805

[20] Porter M. F., An algorithm for suffix stripping, Program, 1980, 14(3), 130–137

[21] Sanderson M., Word Sense Disambiguation and Information Retrieval, In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, (Dublin Ireland), Springer Verlag, 1994, 142–151

[22] Shaw J. A., Fox E. A., Combination of Multiple Searches, Overview of the Third Text REtrieval Conference (TREC-3), NIST Gaithersburg, 1995, 105–108

[23] Voorhees E. M., Using WordNet to disambiguate word senses for text retrieval, In: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, (Pittsburgh USA), ACM New York, 1993, 171–180

[24] Voorhees E. M., Harman D, Overview of the Seventh Text REtrieval Conference (TREC-7), NIST Gaithersburg, 1998