Combining linguistic indexes to improve the performances of information retrieval systems: a machine learning based solution

Fabienne Moreau, Vincent Claveau and Pascale Sébillot
IRISA
Campus universitaire de Beaulieu
35042 Rennes cedex, France

{Fabienne.Moreau,Vincent.Claveau, Pascale.Sebillot}@irisa.fr

Abstract

Taking into account in one same information retrieval system several linguistic indexes encoding morphological, syntactic, and semantic information seems a good idea to better grasp the semantic contents of large unstructured text collections and thus to increase performances of such a system. Therefore the problem raised is of knowing how to automatically and efficiently combine those different information in order to optimize their exploitations. To this end, we propose an original machine learning based method that is able to determine relevant documents in a collection for a given query, from their positions within the result lists obtained from each individual linguistic index, while automatically adapting its behavior to the characteristics of the query. The different experiments that are presented here prove the interest of our fusion method that merges the result lists, which obtains better overall and also more stable results than those got by the better individual index.

1 Introduction

Information retrieval systems (IRSs) aim at establishing a relation between users' information needs (generally expressed by natural language queries) and the information contained in documents. To this end, a very common method consists in representing the content of documents and queries as (weighted) sets of words. In this framework, a document is said to be relevant for a query if it shares a certain amount of terms with it. With such a mechanism, IRSs face two problems, mainly bound to the inherent complexity of natural language. The first one is related to polysemy: a same word can have different meanings and represent several concepts (*e.g.* `bug`: `insect` or `computer problem`); because of such ambiguities, IRSs may retrieve non relevant documents. The second and dual issue reflects the fact that a same idea can be expressed by different forms (*e.g.* `bicycle-bike`). Therefore, a relevant document can contain terms semantically close to those of the query but graphically different, and such a document will not be retrieved by standard IRSs.

In order to better grasp the semantic contents of documents and to overcome those two previously mentioned difficulties —especially critical in the case of large textual collections in which those phenomena are pronounced—, a quite obvious solution is to perform a linguistic analysis of both documents and queries, using natural language processing (NLP) techniques.

This allows one to obtain richer and more robust descriptors than simple strings of characters, thus making a more relevant document-query matching possible. Indeed, these descriptors should be able to highlight the fact that a same word can have different meanings or undergo variations of form (`retrieve` ↔ `retrieval`), structure (`information retrieval` ↔ `information that is retrieved`) or meaning (`seek` ↔ `search`).

Many previous researches have tried to enrich IRSs with different kinds of linguistic information. However, they have often resulted in disappointing, unclear, and even sometimes contradictory conclusions. In order to obtain more significant results concerning the contribution of linguistic information in information retrieval (IR), we propose here a new approach for coupling NLP and IR. In contrast with the studies that generally handle only one type of linguistic knowledge, we choose to make the most of the richness of language by combining several levels of linguistic information through morphological, syntactic and semantic analyses. We make the assumption that the combination of those multilevel information should offer a richer characterization of the textual contents and consequently contribute to improve the performances of IRSs, offering a deeper semantic access to contents.

Consequently, the underlying challenge is to optimally exploit those various pieces of linguistic information. Indeed, they do not always all have the same impact on performances. Moreover, some of them are complementary to retrieve relevant documents while on the contrary others are redundant. Furthermore, integrating these linguistic pieces of information in a single index[1] is a thorny issue: how does one find an homogenous way to represent these different pieces of information in a single data structure? how does one weight their relative importance without a priori? In order to overcome these problems, one approach consists in building one separate index for each type of linguistic knowledge extracted from the documents. These indexes are then used independently by an IRS and, for a given query, their results are merged. Within such an approach, this merging step is crucial; to be efficient, it needs to adapt automatically its behavior to the respective efficiency of each separate considered linguistic information. To this end, in this paper, we propose a new technique to merge the lists of documents produced by several linguistic indexes (each one corresponding to one type of linguistic information) integrated in parallel within an IRS. Our technique is built on top of a supervised machine learning system that automatically selects the most efficient linguistic information to find relevant documents, taking into account for that aim some query characteristics. This machine learning system on which this paper focusses on provides an original and effective fusion of linguistic information, and results obtained prove the interest of combining within an IRS different kinds of linguistic knowledge.

The remaining of the paper has the following structure: Section 2 presents some related studies; Section 3 describes the system of supervised machine learning that we have developed to merge the result lists of the linguistic indexes. Section 4 details the experiments that have been conducted, presents and analyzes their results, and compare them to those of an efficient system. Finally, Section 5 concludes on the relevance of our system to improve the performances of IRSs.

[1] Following the tradition in textual IR, we use the term `index` as a synonym for `descriptor`.

2 Context and related work

In this section, we present the context of our work and some existing studies that are related to our problematics. We focus more precisely on the use of linguistic information in IR, and on the problem of data fusion.

2.1 NLP-IR coupling

NLP tools and techniques are generally used in IR to create richer representations of documents and queries in order to provide a more relevant matching between them. As mentioned in introduction, many studies have already tried to use linguistic information in IR. Generally, only one type of analysis is performed: a morphological, syntactic or semantic one. Morphological information, often obtained through a process of stemming or lemmatization (Fuller and Zobel, 1998), can help IRSs to recognize within documents and queries the different forms of a single word and match them. As an example, a query with the term `knife` can be matched with a document containing `knives`. Syntactic information, *e.g.* complex terms or noun phrases (Perez Carballo and Strzalkowski, 2000; Fagan, 1987, *inter alia*), offers the advantage in IR to take into account relations and dependencies that terms share. Thus, it makes it possible to exceed weaknesses of the so-called traditional representation in bags of words. Last, the integration of semantic analyses in IRSs can contribute to improve their performances while seeking, for instance, to associate each document and query with a set of non ambiguous meanings (*cf.* for example (Kilgarriff and Palmer, 2000; Sanderson, 1997)) or to add terms semantically related to the words initially contained in the query (Gauch et al., 1999; Voorhees, 1998).

Despite the number and the diversity of the studies already conducted in the past, it remains difficult to draw up a precise assessment of the contributions of these linguistic information in IR. Conclusions are often contradictory and obtained results depend on numerous parameters, like the language processed, the length of the queries, or the size of the collection. One explanation of those mixed results is related to the fact that most of the existing studies generally integrate linguistic information belonging to only one level of language. Consequently, they only partially take into account the richness of the language since morphological, syntactic and semantic levels are dependent on each other. Some rare studies (Strzalkowski et al., 1996) have already proposed to fully exploit the diversity of the knowledge extracted by NLP tools by simultaneously taking into account those three levels of linguistic analysis. However, performed experiments have concluded that linguistic information can contribute to improve IRS results but only in a very modest way (*e.g.* Strzalkowski *et al.* reports a +5% improvement of 11-pt average precision with his system combining different linguistic representations).

More recent work (Moreau, 2006) shows the potential benefit of combining multilevel linguistic representations in IRSs. Indeed, a thorough analysis of the correlations and relations among several multilevel linguistic information has highlighted interesting cases of complementarities between these information and more particularly between morphological and semantic information to detect relevant documents. These encouraging results allow us to postulate the relevance of combining various linguistic representations in IR. In this paper, we propose to experiment these ideas by integrating in parallel within a same IRS 12 indexes, each one resulting of a different linguistic analysis of the documents (see details in Section 3.1). Thus, the problem raised is of knowing how to automatically and efficiently combine these different pieces

of knowledge within the IRS in order to optimize their exploitations. To this end, we propose to merge the result lists (*i.e.*, the lists of documents ranked by descending order of relevance for a given query) obtained by each linguistic descriptor corresponding to one type of linguistic information. This fusion provides a final list of results corresponding to the best documents found by the indexes. Data fusion is nevertheless quite a complex problem, that is well-known in IR.

2.2 Data fusion in IR

Data fusion has been exhaustively investigated in the literature, especially in the framework of IR (for a state-of-the-art, see (Croft, 1997)). The difficulty is to find a way to combine results of multiple searches conducted in parallel on a common data set for a given query in order to obtain higher performances than each individual search. Each search produces an ordered list containing the documents found to be relevant for a query. One document is associated on the one hand with a relevance score (that represents its degree of similarity to the query) and on the other hand with a rank corresponding to its position in the list. Different techniques have been proposed for the fusion of these lists. Some are based on the relevance scores associated with the documents. They generally sum all the normalized relevance scores obtained by one document retrieved by several systems (or present in various result lists) (*e.g.* combSUM algorithm (Fox and Shaw, 1994)) or multiply this sum by the number of lists that contain the document (*e.g.* combMNZ algorithm (Bartell et al., 1994; Lee, 1997, *inter alia*)). They get a final score for each document from which a ranking can be obtained.

However, the relevance scores provided by the different searches can sometimes be too dissimilar to be merged easily. This is one of the reasons that has motivated other data fusion techniques, based on the rank information. These methods consider the data fusion problem as a multi-candidate election (where the documents are the candidates and the lists of results the voters), and use rank-aggregation algorithms like the Borda or Condorcet counts (Aslam and Montague, 2001). All these methods perform in an unsupervised way: they only use the information retrieved by systems for the fusion. Their problem is that non-relevant documents that are present in several lists are likely to obtain an important weight and to be well ranked in the final list. To overcome these limitations, supervised data fusion techniques (Vogt and Cottrell, 1999; Perez Carballo and Strzalkowski, 2000) calculate the final relevance score of a document as a linear combination of its normalized scores within the different lists, themselves weighted according to their efficiency to detect the relevant documents. The importances given to the lists are fixed *a priori* (this is the supervised aspect of those approaches), and are generally based on the individual performances of each system evaluated with the help of relevance judgments.

Within the framework of our work, those classical methods for data fusion are problematic for several reasons. First, with such traditional techniques, documents that are present in a high number of result lists are favored and can get an important weight in the final result list. However, in our case, a relevant document can be retrieved by only one index (representing one particular linguistic information; all the other linguistic pieces of information being unable to detect the relevant relation between the query representation and the document one). During the fusion, a strong importance must consequently be given to this document. Moreover, in our case, the performances of the various linguistic indexes are very dissimilar (for example, indexes based on morphological information are often more performant than syntactic ones).

Thus, an identical importance cannot be given to the different lists; indexes that appear more efficient to improve the performances of IRSs must be favored. However, we do not want to give *a priori* more importance to the indexes that seem the best ones. Indeed, their efficiency may vary from one query to another, and some indexes that are found to be often less effective can sometimes be powerful to retrieve relevant documents. Therefore we need a flexible and adaptative fusion method. At last, one of our assumptions is that the efficiency of an index depends on the type of the considered query, and more particularly on the nature of the linguistic information it contains. It appears quite natural for instance that for a query in which a proper name appears, an index about proper names is privileged compared to others. Conversely, for a query including only common nouns, this index should have less impact. Thus, it is necessary to also adapt the weights of the index results to the characteristics of the query. Many studies in IR claims that the quality of the results is strongly dependent on the considered query. Most of these studies aim at predicting the difficulty of queries and at estimating the reliability of the results obtained by an IRS that processes them (Mothe and Tanguy, 2005; Macdonald et al., 2005; Cronen-Townsend et al., 2002). As a whole, experiments have proved that there is a strong correlation between information that characterize a query —numerical features (query term frequency in the collection of documents for instance) or more symbolic ones (*e.g.* the number of senses of an ambiguous term, the presence or absence of proper names...)— and performances of IRSs.

In this context, we propose to conceive a flexible method for the fusion of the results that takes into account query characteristics in order to automatically detect the best way of combining result lists. To this end, we use a supervised machine learning technique, namely neural networks, to learn to automatically evaluate the relevance of documents for a given query according to their positions in the different result lists and to linguistic information about the query. Finally, documents that are found to be relevant by the inferred neural network are merged in a unique result list.

3 A supervised machine learning system to merge result lists

In this section, we first give an overview of the linguistic elements that we take into account to describe the semantic contents of both documents and queries. Then, we describe our machine learning approach used to combine the result lists produced by the various linguistic indexes.

3.1 Content description

Several linguistic analyses are performed on the considered collection of documents and queries (see Section 4.1 for the description of the collection used in our experiments). These analyses are based on *common* NLP techniques and tools that automatically extract from the documents and queries 11 different types of linguistic information latter used independently to build the different indexes. Our aim is actually not to try to obtain "perfect" linguistic information but to show that combining information extracted from large collections by standard tools, when realized in a relevant way, offers a better semantic access to contents and improve performances of IRSs. The linguistic pieces of information are the following ones:

- morphological information: lemma (a word without its inflections (gender, tense, number or person); *e.g.* `companies` is transformed into `company`), stem (*e.g.* `compiler`,

`recompiling` are both transformed into `compil`), grammatically tagged term (depending on its context, the ambiguous word form `drink` is tagged as a singular noun or a verb);

- syntactic information: complex term (*e.g.* `neural network` that have a more precise meaning than `neural` + `network`), noun phrase (*e.g.* `information that is retrieved` that can be represented by a head-modifier relation `retrieve+information`), bigrams (sequences of two following words), trigrams (sequences of three following words);

- semantic information: semantically tagged term (a term associated with the number of its senses in the well-known lexical thesaurus WORDNET (Fellbaum, 1998)), term + set of synonyms (a term associated with the set of its synonyms extracted from WORDNET), term + morphological and semantic variants (a term associated with a set of words related by a link of derivational morphology in WORDNET), proper names.

Each type of extracted linguistic information is used to build a different index. The linguistic elements are weighted according to the BM-25 formula. Thus, each document (or query) of the collection is represented by 11 different descriptors: a document can be seen as a (weighted) set of lemmas, stems, grammatically tagged terms, complex terms, noun phrases, bigrams, trigrams, proper names, semantically tagged terms, terms associated with a set of synonyms, terms associated with a group of morphological and semantic variants. Documents and queries are also represented by a classical index: the set of the simple terms that they contain. Finally, 12 indexes are used to represent their textual contents. These 12 various document and query representations are then integrated in a parallel way within an IRS. In the experiments presented below, we have chosen to use LEMUR[2], configured to emulate the well-known OKAPI IRS. Each document representation is compared with its corresponding query representation, and a similarity score is computed. This matching phase enables us to obtain for each index a document list ranked by decreasing order of relevance for a given query. Finally, 12 ordered result lists are generated; these 12 lists are the ones that will be considered for the fusion process using the machine learning technique approach described in the following sub-section.

3.2 Principles and methods

The automatic merging of the result lists is tackled as a supervised classification problem with 2 classes. The goal is to determine if each document of the collection has to be considered as relevant or not relevant for a given query, taking into account its ranking within the different result lists and some information about the query. All the documents considered as relevant will form the final list of results. In order to perform this fusion, our approach proceeds in 2 steps: a training phase, during which a classifier is inferred, and a utilization phase which consists in using the classifier on new queries and documents. The format of the input data common to both the classifier in its utilization phase and the machine learning system in its training phase (learning the classifier) except for the presence of relevance judgments, is described below; the training and utilization steps are then presented in sub-section 3.2.2; more technical precisions about the inference of the classifier and justifications of the choice of neural networks as a machine learning system are given in sub-section 3.2.3.

[2] LEMUR is available at the following URL: `http://www.lemurproject.org/`

3.2.1 Input data

Input data of our system are query-document pairs (each document of the collection is evaluated for a given query) that are characterized by several attributes. Our approach rely on these attributes to determine, by using the machine learning system, if the document is relevant for the given query and thus if it belongs to the final list of results. Two types of attributes can be distinguished to characterize such a pair: the attributes used to describe documents and those used to represent queries.

Concerning the document representation, for each query, we keep the ranks of each document in the result lists provided by the 12 indexes. If the document does not belong to one of the 12 lists (this is the case if, for a particular query, the document was not retrieved by Lemur using the considered index), the value we keep as attribute is set to zero.

Concerning the query characterization, some simple features are directly extracted from the queries. These features are those that are found to be efficient in existing studies about the prediction of query difficulty (*cf.* Section 2.2). About thirty elements are chosen. First, we take into account the influence of the length of a query by computing its size (*i.e.*, number of words), its number of sentences, its number of full words, etc. We also use various linguistic information contained in the query: morphological information (number of simple terms, lemmas, stems, grammatically tagged terms in the query), syntactic information (number of verbs, bigrams, noun phrases, complex terms, trigrams in the query) and semantic information (number of proper names, disambiguated terms, average number of meanings, number of non-ambiguous terms in the query). Finally, we benefit from information about the query specificity relying on the frequency of its terms: frequency of the linguistic information in the query (average frequency of the simple terms, lemmas, stems, grammatically tagged terms, bigrams, noun phrases, complex terms, proper names in the query) and in the document collection (average documentary frequency of the simple terms, lemmas, roots, bigrams, noun phrases, complex terms, proper names contained in the query). Those elements are automatically extracted using the same NLP tools and resources used to build the 12 indexes. Some of these characteristics are not available for all the queries (they do not all contain proper names for instance); a zero value is then associated with the missing characteristics.

All these pieces of information constitute the different attributes of a query-document pair that is given as input to our system. Such a pair can be seen as a vector (noted $\overrightarrow{query - doc}$) composed of n components $x_1, ...x_i, ..., x_n$ that correspond to the set of attributes used for its characterization. Table 1 presents all those attributes.

During the training phase, since we are using a supervised machine learning technique, each query-document pair is also associated with the expected class value wished as output (document relevance or non relevance). Finally, examples used for the training step are noted as pairs like: $(\overrightarrow{query - doc},\ relevance\ decision)$.

3.2.2 System architecture

The global organization of the system we propose for the fusion of our result lists is relatively straightforward. It is modelled on the common training/testing framework used for machine learning problems. Figure 1 illustrates this process.

x_1	document rank retrieved by the simple term index
x_2	document rank retrieved by the lemma index
x_3	document rank retrieved by the stem index
x_4	document rank retrieved by the grammatically tagged term index
x_5	document rank retrieved by the noun phrase index
x_6	document rank retrieved by the bigram index
x_7	document rank retrieved by the complex term index
x_8	document rank retrieved by the trigram index
x_9	document rank retrieved by the proper name index
x_{10}	document rank retrieved by the semantically tagged term index
x_{11}	document rank retrieved by the synonym index
x_{12}	document rank retrieved by the semantically and morphologically related word index
x_{13}	number of sentences in the query
x_{14}	query length (number of words)
x_{15}	number of full words in the query
x_{16}	number simple terms in the query
x_{17}	number of lemmas in the query
x_{18}	number of stems in the query
x_{19}	number of grammatically tagged terms in the query
x_{20}	numebr of verbs in the query
x_{21}	number of bigrams in the query
x_{22}	number of noun phrases in the query
x_{23}	number of complex terms in the query
x_{24}	number of trigrams in the query
x_{25}	number of proper names in the query
x_{26}	number of disambiguated terms in the query
x_{27}	average number of senses in the query
x_{28}	number of non ambiguous terms in the query
x_{29}	simple term average frequency in the query
x_{30}	lemma average frequency in the query
x_{31}	stem average frequency in the query
x_{32}	grammatically tagged term average frequency in the query
x_{33}	bigram average frequency in the query
x_{34}	noun phrase average frequency in the query
x_{35}	complex term average frequency in the query
x_{36}	proper name average frequency in the query
x_{37}	query simple term average documentary frequency
x_{38}	query lemma average documentary frequency
x_{39}	query stem average documentary frequency
x_{40}	query bigram average documentary frequency
x_{41}	query noun phrase average documentary frequency
x_{42}	query complex term average documentary frequency
x_{43}	query proper name average documentary frequency
x_{44}	query trigram average documentary frequency

Table 1: Query-document vector components used as input data

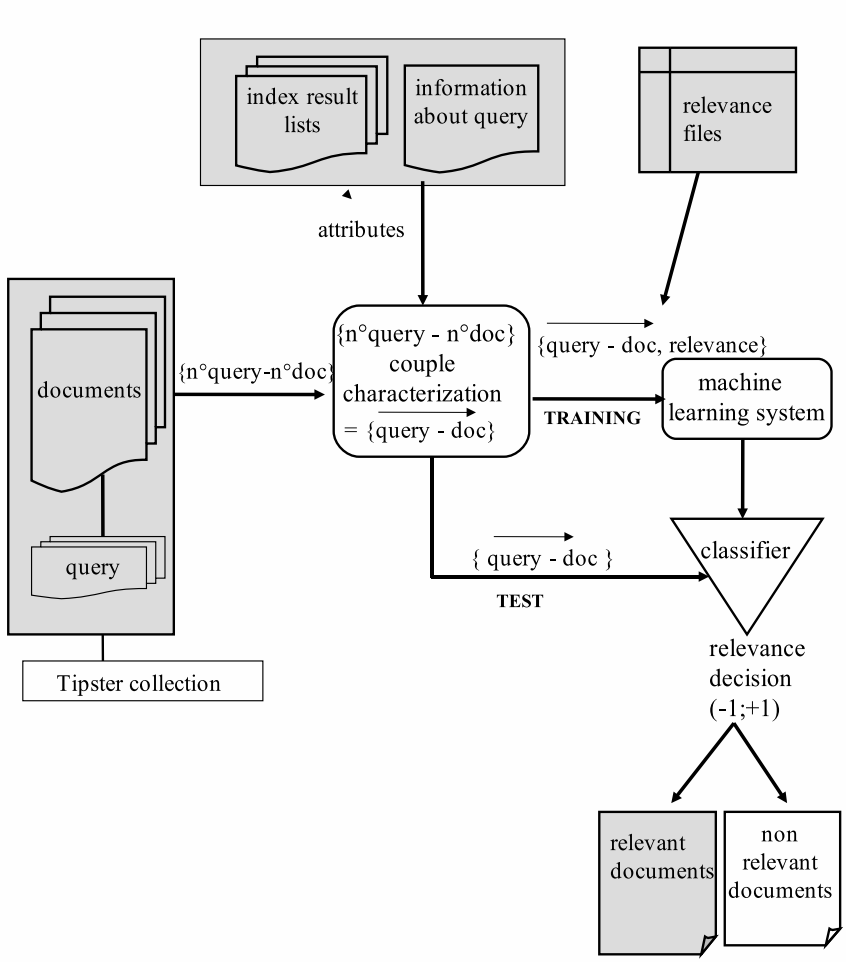

Figure 1: System architecture proposed for the fusion of the result lists

As previously mentioned, our fusion method is based on a system that is composed of two stages. The first one is the training phase that consists in giving as input to a neural network a set of example-couples. Each example corresponds to a vector represented by a set of attributes, associated with a relevance decision. From these couples and their attributes, the neural network tries to learn how to distinguish the relevant documents from non relevant ones for a given query. When this training is finished, the resulting neural network can be used as a classifier. If the learning has been correctly performed, the classifier is able to automatically establish the relevance (or non relevance) of each document of the collection for a given query from its positions within the different result lists and taking into account information about the query, and thus to build a set of all the documents finally found relevant for the query. Note that the inferred neural network does not give a score to documents, but only indicates if they are relevant or not for a given query (*i.e.*, binary decision).

In order to evaluate the performances of the classifier and more generally to measure the whole system efficiency, unseen query-document pairs associated with their characterizing attributes are given to the classifier. The classifier provides a decision on their relevance that can be compared with a manually given relevance judgment.

The efficiency of our method strongly depends on the quality of the input data. The attributes

used to describe query-document couples must be sufficiently discriminating and relevant to enable the system to differentiate relevant documents from non relevant ones for any query. The assumption made here is that the rank information issued from the 12 linguistic indexes and the query attributes are sufficiently reliable for the system to effectively perform the fusion task. Performances of our approach are also dependent on the quality of the machine learning system, and thus, on the neural network ability to learn and generalize the examples in order to build an effective classifier.

3.2.3 Inferring the classifier

The machine learning technique used to perform the classification task described above is the neural network inference. Among all the possible machine learning technique, neural networks have been chosen for different reasons. First, neural networks have proven to be very efficient on numerous classification problems, and many well-documented softwares are available. Secondly, all the attributes are expressed as numerical information; and neural networks are well-suited to handle such types of data. Thirdly, it is well worth noting that our fusion system must be able to manage a high number of data. Indeed, in order to obtain the final list, the system must predict the relevance of each collection document for each of the queries, using 44 attributes to describe each query-document couple. Thus, the classifier used to process the data has to be relatively fast; this is the case of neural networks. Last, neural networks are quite tolerant with noisy data. This is an important criteria to take into account since the attributes used to describe the examples are automatically extracted using NLP techniques with sometimes low performances.

In practice, in the experiments presented below, the neural network implementation we use is the open-source software FANN[3]. The training step of the neural network takes a few minutes on a standard desktop PC. Once this inference step is done, classifying with the resulting neural network all the 175,000 documents of the collection (see below) for a given query takes less than a second.

4 Experimental results

This section is dedicated to a set of experiments that have been performed to evaluate the efficiency of our fusion method applied on results obtained by the different linguistic indexes, and consequently to estimate the interest of combining several linguistic information within an IRS to better grasp the semantic contents of documents and queries. Sub-section 4.1 presents the IR collection used for our experiments; Sub-section 4.2 describes our methodology to partition data for the classifier training and test to enable evaluation. Finally, Section 4.3 details and analyzes the results obtained by our fusion approach on the selected collection.

4.1 Data description

In order to test our merging approach, we need two sets of data: a training collection used to infer a classifier, and a test collection used to evaluate the performances of the resulting classifier. For both sets, we need queries, documents and the corresponding relevance judgments (list of relevant documents for each query). In our experiments, these necessary data are taken out of a subset of the TIPSTER collection used in TREC. More precisely, we kept a Wall Street Journal

[3] FANN library is available at the following URL: `http://leenissen.dk/fann/`.

subcollection made up of about 175,000 documents, and a set of 50 queries (from TREC-3) with their relevance judgments have been retained.

The whole set of documents has been analyzed by NLP tools and the 12 linguistic indexes have been built according to the process presented in Section 3.1. For each query, our IRS performs 12 runs, each one using a different index. We obtain 12 result lists containing all the documents retrieved, ranked in descending order of their relevance for a given query. To feed up the neural network, we also need query characteristics; they are obtained on each of the 50 queries with the help of NLP tools and resources as it is described in Section 3.2.1.

4.2 Methodology for the evaluation

In order to effectively estimate the performances of our classifier, two conditions have to be met. First, it is necessary to test the classifier on a data sample that is independent from the training dataset. Secondly, repeating the training/test operations makes the evaluation results more reliable. To bring these two conditions into operation, we rely on the k-fold cross validation method. With this evaluation technique, commonly used in the machine learning community, the original data sample is partitioned into k subsamples. From those k subsamples, a single subsample is retained as an evaluation data set to test the classifier, and the $k-1$ remaining subsamples are used together as training data. Then, the cross-validation process is repeated k times, each of the k subsamples being used exactly once as the validation data set. The k results from the folds can then be averaged to produce a single classifier estimation.

In our case, we perform a 10-fold cross validation: the original sample data (*i.e.*, the 50 queries with their relevance judgments) is randomly partitioned into 10 subsamples. Each of these subsamples is composed of 5 queries that are used alternatively as test set to evaluate the training performed from the 9 other subsamples (*i.e.*, 45 queries). The classifier overall performance is the average of the performances obtained on each of the 10 test subsamples.

For each test set, we evaluate if the decision of relevance provided by the classifier for each document retrieved for a given query corresponds to the relevance indicated in the relevance files. This process is repeated for the 5 queries of a test set. For this evaluation, the main measure used is the F-measure (Rijsbergen, 1979) that combines recall and precision in a single efficiency measure (here with an identical weight given to precision and recall). To compute the F-measure value, we also calculate the recall and precision rates obtained for each query. Let us notice that result lists that are evaluated (*i.e.*, documents that have been classified relevant for a given query by the classifier) are not ordered, as previously mentioned; only a relevance binary judgment associated with the documents for each query is obtained. This constraint prevents us from using other evaluation measures commonly used in IR, such as the non-interpolated average precision (MAP) for example.

4.3 Results and discussion

Several experiments have been performed in order to evaluate our fusion approach and thus the interest of combining several linguistic information in a single IRS. First, the overall performances of our classifier is evaluated and compared with results of a traditional IRS. Secondly, in order to understand the observed results, a thorough study of the performances obtained on

each evaluated query is proposed. Finally, a last experiment is carried out to evaluate the impact of taking into account information about the queries in our fusion method.

4.3.1 Overall evaluation

For each query of one given test set, the inferred classifier produces a non-ordered result list. In order to evaluate its overall performance and, consequently, to estimate the relevance of the retrieved documents, we calculate the F-measure (harmonic mean of precision and recall) averaged for each of the 10 test sets. The results of our merging method are compared to those obtained on the same collection by the individual index observed as the most efficient among our 12 indexes, *i.e.*, the stem index. Moreover, for this comparison, we want to confront ourselves with the most difficult case. Thus, our performances are compared to the best results obtained by this stem index, evaluated with the DCV (document cut off value) giving the highest F-measure value (DCV=100 in the following experiments). Table 2 summarizes the averages of the precision, recall and F-measure values that are obtained on the 10 test sets by the 2 IRSs: the IRS that integrates only stems and our neural network based system.

	Stem index (DCV=100)	Fusion method (improvement %)
Precision	29.70	29.48 (-0.73%)
Recall	50.80	43.88 (-13.61%)
F-measure	30.53	34.30 (**+12.33%**)

Table 2: Precision, recall and F-measure averages obtained by the fusion method and compared to the performances of a stem-based IRS

Reported figures show the overall efficiency of our result list fusion method. Indeed, good results for the F-measure are obtained since a relative improvement of 12% is observed compared to the stem index performances. By comparing the efficiency of our approach with performances of a IRS only based on the best index (stem index), we have proved that our fusion method does not only rely on the results of the most efficient index but also benefits from the other indexes and from query information to propose better results. Combining linguistic knowledge, provided it is realized in a relevant way is thus interesting to better access to the semantic contents.

Nonetheless, Table 2 also shows that the results obtained by our approach are overall identical in precision but lower in recall, which is surprising at first glance. The fact that the F-measure value in our case is higher seems to indicate that our method offers more balanced precision-recall compromises than the stem-based IRS. In order to test this assumption, the performances of both IRSs for each considered query are detailed in next experiments.

4.3.2 Performances query by query

A query by query manual analysis of the results of the 2 IRSs enables us to notice that performances observed by the sole stem index strongly vary from one query to another. For some queries, the stem-based IRS yields very good results for the recall measure (close to 100%) but generally associated with a very low precision (close to 5%) which leads to an average recall higher than ours but a very low F-measure (about 10). More generally, the results of the stem

index appear more irregular than those provided by our method. In order to confirm this first idea, we detail the results obtained on each test set: averages and standard deviations of the precision, recall and F-measure values obtained on the 10 test sets are computed for the two systems. The standard deviation gives an strong indication of the dispersion of the results for the different queries of a test set compared to the computed average. These informations are summarized in tables 3, 4 and 5.

Test data partitioning (cross-validation)	Stems (DCV=100) F-measure average (standard deviation)	Fusion F-measure average (standard deviation)
Test set 1	45.64 (13.32)	46.55 (5.61)
Test set 2	23.33 (11.26)	24.81 (5.82)
Test set 3	30.43 (16.67)	36.72 (4.42)
Test set 4	21.68 (18.32)	32.51 (3.33)
Test set 5	28.13 (18.32)	40.49 (4.25)
Test set 6	33.34 (15.74)	34.17 (4.04)
Test set 7	29.76 (20.55)	30.25 (3.29)
Test set 8	26.79 (11.87)	27.40 (2.12)
Test set 9	40.67 (17.20)	43.98 (1.64)
Test set 10	25.54 (17.68)	26.10 (4.70)
Average on the 10 test sets	30.53 (16.09)	34.30 (3.92)

Table 3: **F-measure** averages and standard deviations on the 10 test sets of the stem index and the fusion method

Test data partitioning (cross-validation)	Stems (DCV=100) Precision average (standard deviation)	Fusion Precision average (standard deviation)
Test set 1	49.59 (24.49)	47.47 (4.37)
Test set 2	18.11 (11.73)	18.28 (4.38)
Test set 3	23.54 (15.30)	28.37 (3.71)
Test set 4	19.95 (21.15)	25.23 (3.97)
Test set 5	43.30 (22.92)	36.52 (4.09)
Test set 6	26.59 (14.13)	25.98 (4.17)
Test set 7	23.67 (18.07)	24.16 (3.41)
Test set 8	22.07 (13.33)	19.08 (1.96)
Test set 9	37.62 (25.93)	40.16 (3.83)
Test set 10	32.31 (26.15)	29.51 (4.26)
Average on the 10 test sets	29.70 (19.32)	29.48 (3.81)

Table 4: **Precision** averages and standard deviations on the 10 test sets of the stem index and the fusion method

Test data partitioning (cross-validation)	Stems (DCV=100) Recall average (standard deviation)	Fusion Recall average (standard deviation)
Test set 1	47.57 (13.41)	43.18 (7.16)
Test set 2	47.13 (13.65)	38.86 (9.70)
Test set 3	61.21 (11.32)	52.33 (6.89)
Test set 4	55.32 (32.49)	46.51 (2.44)
Test set 5	40.31 (18.60)	45.52 (5.11)
Test set 6	52.57 (26.11)	50.93 (7.68)
Test set 7	49.03 (29.66)	40.75 (3.36)
Test set 8	41.73 (7.85)	47.57 (3.95)
Test set 9	75.53 (25.99)	49.50 (6.07)
Test set 10	37.59 (21.10)	23.63 (5.36)
Average on the 10 test sets	50.80 (20.02)	43.88 (5.77)

Table 5: **Recall** averages and standard deviations on the 10 test sets of the stem index and the fusion method

Very significant differences between the results obtained by the 2 IRS are shown by the figures in the tables, as well in terms of precision, recall or F-measure. Concerning our method, the standard deviation observed on the 10 test sets is very low since obtained values vary between 1.64 and 5.82 for F-measure, between 1.96 and 4.38 for precision, and between 2.44 and 9.70 for recall. Consequently, obtained results on each query are very close to the average. Results are also constant whatever the query. For the stem-based IRS, the standard deviation is much higher. The values vary between 11.26 and 20.55 for F-measure, between 11.73 and 26.15 for precision, and between 7.85 and 32.49 for recall. These latest figures clearly attest of a more important dispersion of the results for the different queries around the average.

The stem index makes very significant improvements for some queries but appears less effective for others by providing unbalanced results (high recall and low precision or conversely). The higher stability of our results proves the capacity of our method to compensate for cases when, for a given query, the stem index fails, taking advantage of the other indexes. Therefore, the main contribution of our method is to perform a smoothing of the results from the different indexes and to make them consequently less sensitive to the types of the queries. Since the observed results are constant on the various evaluated queries, an assumption according to which the system has adapted its behavior to the specificities of the queries to select the indexes that are likely to be the most effective to retrieve the relevant documents can be made. In order to confirm this idea, the following experiments propose to evaluate the influence of taking into account information about the query on the efficiency of our fusion method.

4.3.3 Impact of query characterization

In order to validate the assumption that our fusion method bases its behavior on characteristics of the queries to select the best indexes, we conduct a simple additional experiment. We repeat the previous experiments, removing from our system all the attributes that correspond to information about the queries (*i.e.*, attributes x_{13} to x_{44}). In other words, our neural network only learns from information about document ranks. Comparing results of the two experiments

(*i.e.*, performances with and without taking into account information about queries) allows us to have a precise idea of the benefit one can expect of our method.

Table 6 presents the precision, recall and F-measure average values obtained on the 10 test sets by the IRS that integrates stems and by the fusion system without information about queries. The reported figures clearly show that our classifier performs not so good when it cannot use query features. Indeed, the F-measure improvement is low, and both precision and recall are lower than the ones obtained by the stem-based IRS. Nonetheless, as for the previous results, the fact that the F-measure of the fusion method is higher than that of the stem IRS while precision and recall are lower seems to indicate that our system still provide more stable compromises.

	Stem index	Fusion (without query info.) (improvement %)
Precision	27.90	26.49 (-5.04%)
Recall	51.32	41.65 (-18.84%)
F-measure	29.67	30.28 (+2.05%)

Table 6: Precision, recall and F-measure averages obtained by the fusion method without query information and compared to the performances of the stem-based IRS

Once again, we detail the results query by query to observe the variations. Tables 7, 8 and 9 present respectively F-measure, precision and recall values for each of the 10 test sets. First, these results show again that results obtained by our method without any information about the queries are clearly not as good as the previous ones. These observations highlight the idea that our fusion approach benefits a lot from the query features to identify the indexes likely to be most effective to find the relevant documents. Thus, our approach is more complex than traditional ones that are only based on result lists, but is also more flexible since it is able to adapt differently its behavior to each type of query. These figures also prove that exploiting jointly linguistic information from the morphological, syntactic and semantic levels of language in IR is of great interest since results are more stable than those observed in the NLP-IR coupling experiments that take into account only one linguistic information and whose performances are known to be very irregular. Yet, these latest experiments also prove that our fusion system needs information about queries in order to improve the overall performances (in terms of F-measure).

5 Conclusion and future work

In order to make the best possible use of rich linguistic representations of documents and queries in IRSs, representations better suited to a semantic access to contents, an automatic technique is required, able to take those representations into account and combine their elements in the most adapted way. This paper described our solution to that issue. We propose an original method for data fusion, based on a supervised machine learning technique, namely neural networks, that is able to determine the relevance of a document for a given query by considering the positions obtained by this document for this query in different result lists corresponding to several linguistic indexes of various levels of language (morphological, syntactic

Test data partitioning (cross validation)	Stems	Fusion (without query info.)
	F-measure average (standard deviation)	F-measure average (standard deviation)
Test set 1	17.15 (15.60)	19.89 (4.09)
Test set 2	45.23 (17.06)	46.67 (4.50)
Test set 3	23.65 (14.42)	24.58 (3.16)
Test set 4	38.42 (11.71)	37.18 (4.89)
Test set 5	26.55 (16.04)	24.37 (1.74)
Test set 6	30.32 (12.66)	22.65 (3.07)
Test set 7	21.58 (13.88)	24.14 (4.73)
Test set 8	31.21 (20.90)	42.62 (8.57)
Test set 9	27.52 (13.46)	28.32 (1.90)
Test set 10	35.07 (19.92)	32.38 (3.03)
Average on the 10 test sets	29.67 (15.57)	30.28 (3.97)

Table 7: **F-measure** averages and standard deviations on the 10 test sets of the stem index and the fusion method without query information

Test data partitioning (cross-validation)	Stems	Fusion (without query info.)
	Precision average (standard deviation)	Precision average (standard deviation)
Test set 1	13.73 (16.99)	12.53 (3.02)
Test set 2	45.91 (20.07)	41.07 (6.02)
Test set 3	18.41 (9.79)	17.78 (3.29)
Test set 4	36.70 (22.10)	33.50 (4.73)
Test set 5	35.05 (26.06)	35.18 (4.69)
Test set 6	22.34 (11.24)	15.62 (2.22)
Test set 7	16.53 (13.85)	15.99 (3.68)
Test set 8	33.98 (30.41)	39.69 (7.14)
Test set 9	21.90 (11.64)	21.74 (1.75)
Test set 10	34.48 (25.90)	31.84 (3.49)
Average on the 10 test sets	27.90 (18.81)	26.49 (4)

Table 8: **Precision** averages and standard deviations on the 10 test sets of the stem index and the fusion method without query information

and semantic) and linguistic information about the query. From a computing point of view, it is well worth noting that this architecture is also well suited to process an arbitrary large amount of data since the indexing and inference parts are done off-line, and using the neural network to determine the relevance of a document for a given query is very fast and can be done on-line.

When compared to the same IRS not integrating our fusion method but using only the most

Test data partitioning (cross-validation)	Stems	Fusion (without query info.)
	Recall average (standard deviation)	Recall average (standard deviation)
Test set 1	60.83 (24.23)	49.30 (5.84)
Test set 2	50.78 (11.51)	54.41 (2.52)
Test set 3	40.11 (25.12)	40.53 (3.24)
Test set 4	57.38 (24.45)	42.12 (6.84)
Test set 5	30.13 (15.46)	18.73 (1.09)
Test set 6	62.45 (21.99)	41.70 (6.64)
Test set 7	55.19 (14.44)	49.94 (5.73)
Test set 8	46.87 (26.31)	46.10 (10.53)
Test set 9	47.79 (22.77)	40.68 (1.84)
Test set 10	61.69 (27.03)	32.98 (2.88)
Average on the 10 test sets	51.32 (21.33)	41.65 (4.72)

Table 9: **Recall** averages and standard deviations on the 10 test sets of the stem index and the fusion method without query information

efficient single morphological (stem) representation of documents and queries, results obtained proved the interest of our approach, not in terms of precision and recall as it could be expected, but in a better trade-off between these two measures for each query. They have moreover demonstrated the capacity of our machine learning based approach to adapt its behavior differently to each type of queries. The result stability also shows that the joint exploitation of multilevel linguistic information enables to compensate for the weaknesses of the linguistic information when individually exploited (*i.e.*, variations of results according to data and more particularly according to considered queries).

This paper opens many future prospects that need further consideration. First, the fact that our classifier provides only one binary decision on the document relevance currently constitutes one of the main drawbacks of our fusion method since we cannot fully compare our performances with those obtained by some other IRSs. Improvements of the classifier that consist in giving a score to the documents according to their relevances for a given query is being studied. Secondly, in order to overcome the neural network limitations (its "black box" aspect), other machine learning methods (*e.g.* symbolic methods) could be investigated in order to obtain the same results while offering elements for their interpretation. Currently, we cannot know among the various exploited linguistic information which ones or which combinations of them are the most effective to find the relevant documents for a given query. Last, from a more applicative point of view, using this machine learning approach in a relevance feedback framework is being foreseen. The main idea is to train a neural network, with the same 12 linguistic representations, by interacting with a user during a search. For a given query, the user's judgment on a first list of results is used to train a new classifier that can then be considered to generate a more relevant second list of results; the process can be repeated. Thus, in contrast with the approach presented in this paper, a particular classifier would be especially inferred for a given query and thus may yield good results for it, even if it may not be well-suited to process any other query. More generally, in a mutimedia framework, using this kind of machine learning approach can also be

foreseen as a simple way to integrate each media retrieval results in a unique relevance list.

References

Aslam, J. and Montague, M. (2001). Models for Metasearch. In *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, New-Orleans, USA.

Bartell, B. T., Cottrell, G. W., and Belew, R. K. (1994). Automatic Combination of Multiple Ranked Retrieval Systems. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Dublin, Ireland.

Croft, W. B. (1997). Combining Approaches to Information Retrieval. In Croft, W. B., editor, *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, pages 1–36. Kluwer Academic Publishers.

Cronen-Townsend, S., Zhou, Y., and Croft, W. B. (2002). Predicting Query Performance. In *Proceedings of the 25th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Tampere, Finland.

Fagan, J. L. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods*. PhD thesis, Cornell University, New-York, United-States of America.

Fellbaum, C., editor (1998). WORDNET*: An Electronic Lexical Database*. C. Fellbaum, The MIT Press, Cambridge, Massachussets, United-States of America.

Fox, E. A. and Shaw, J. A. (1994). Combination of Multiple Searches. In *Proceedings of the 2nd International Conference on Text Retrieval (TREC)*, Gaithersburg, United-States of America.

Fuller, M. and Zobel, J. (1998). Conflation-Based Comparison of Stemming Algorithms. In *Proceedings of the 3rd Australian Document Computing Symposium*, Sydney, Australia.

Gauch, S., Wang, J., and Rachakonda, S. M. (1999). A Corpus Analysis Approach for Automatic Query Expansion and its Extension to Multiple Databases. *ACM Transactions on Information Systems*, 17(3):250–269.

Kilgarriff, A. and Palmer, M. (2000). Special Issue on Senseval. *Computers and the Humanities*, 34(1/2).

Lee, J. H. (1997). Analyses of Multiple Evidence Combination. In *Proceedings of the 20th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Philadelphia, United-States of America.

Macdonald, C., He, B., and Ounis, I. (2005). Predicting Query Performance in Intranet Search. In *Proceedings of Predicting Query Difficulty - Methods and Applications Workshop, ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Salvador de Bahia, Brazil.

Moreau, F. (2006). *Revisiter le couplage traitement automatique des langues et recherche d'information*. PhD thesis, University of Rennes 1, Rennes, France.

Mothe, J. and Tanguy, L. (2005). Linguistic Features to Predict Query Difficulty - A Case Study on Previous TREC Campaigns. In *Proceedings of Predicting Query Difficulty - Methods and Applications Workshop, ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Salvador de Bahia, Brazil.

Perez Carballo, J. and Strzalkowski, T. (2000). Natural Language Information Retrieval: Progress Report. *Information Processing and Management*, 36.

Rijsbergen, C. J. v. (1979). *Information Retrieval*. Butterworths.

Sanderson, M. (1997). *Word Sense Disambiguation and Information Retrieval*. PhD thesis, Glasgow University, Scotland.

Strzalkowski, T., Guthrie, L., Karlgren, J., Leistensnider, J., Lin, F., Carballo, J. P., Straszheim, T., Wang, J., and Wilding, J. (1996). Natural Language Information Retrieval: TREC-5 Report. In *Proceedings of the 5th International Conference on Text Retrieval (TREC)*, Gaithersburg, United-States of America.

Vogt, C. C. and Cottrell, G. W. (1999). Fusion via a Linear Combination of Scores. *Information Retrieval*, 1(3):151–173.

Voorhees, E. (1998). Using WORDNET for Text Retrieval. In Fellbaum, C., editor, WORDNET*: An Electronic Lexical Database*. The MIT Press.