Automatic Text Summarization Based on Word-Clusters and Ranking Algorithms

Massih R. Amini, Nicolas Usunier, and Patrick Gallinari

Computer Science Laboratory of Paris 6,
8 Rue du Capitaine Scott,
75015 Paris, France
{amini, usunier, gallinari}@poleia.lip6.fr

Abstract. This paper investigates a new approach for Single Document Summarization based on a Machine Learning ranking algorithm. The use of machine learning techniques for this task allows one to adapt summaries to the user needs and to the corpus characteristics. These desirable properties have motivated an increasing amount of work in this field over the last few years. Most approaches attempt to generate summaries by extracting text-spans (sentences in our case) and adopt the classification framework which consists to train a classifier in order to discriminate between relevant and irrelevant spans of a document. A set of features is first used to produce a vector of scores for each sentence in a given document and a classifier is trained in order to make a global combination of these scores. We believe that the classification criterion for training a classifier is not adapted for SDS and propose an original framework based on ranking for this task. A ranking algorithm also combines the scores of different features but its criterion tends to reduce the relative misordering of sentences within a document. Features we use here are either based on the state-of-the-art or built upon word-clusters. These clusters are groups of words which often co-occur with each other, and can serve to expand a query or to enrich the representation of the sentences of the documents. We analyze the performance of our ranking algorithm on two data sets - the Computation and Language (cmp_lg) collection of TIPSTER SUMMAC and the WIPO collection. We perform comparisons with different baseline - non learning - systems, and a reference trainable summarizer system based on the classification framework. The experiments show that the learning algorithms perform better than the non-learning systems while the ranking algorithm outperforms the classifier. The difference of performance between the two learning algorithms depends on the nature of datasets. We give an explanation of this fact by the different separability hypothesis of the data made by the two learning algorithms.

1 Introduction

With the actual huge and continuously growing online text resources, it becomes necessary to help users get quick answers to their queries. Single document text summarization (SDS) can be coupled with conventional search engines, and help users to quickly evaluate the relevance of documents or to navigate through a corpus.

Automated text summarization dates back to the end of the fifties [14]. Two main ideas have emerged to deal with this task; the first was how a summarizer has to treat a

D.E. Losada and J.M. Fernández-Luna (Eds.): ECIR 2005, LNCS 3408, pp. 142–156, 2005.

huge quantity of data and the second, how it may be possible to produce a human quality summary. Different attempts on the latter have shown that a professional summarization system has to encompass discourse understanding, abstraction and language generation [19]. These processes make the summarization very complex and often intractable for on-line textual documents. To deal with the first point, simpler approaches were explored which consist in extracting representative text-spans, using statistical techniques or techniques based on superficial domain-independent linguistic analyses [9, 29]. For these approaches, SDS can be defined as the selection of a subset of the document sentences which is representative of its content. This is typically done by ranking document text-spans with respect to the similarity measure with a relevant source. Most of the recent work in SDS uses this paradigm. Summarization systems can operate in two modes: generic summarization, which consists in extracting text-spans relevant to the main *topics* of a whole document and query-based summarization, which aims at abstracting the information relevant to a given query. For both approaches, it has been shown that in SDS, extracted text-span units do not always retain their precedence orders in the summary [10]. Usually, sentences are used as text-span units but paragraphs have also been considered [17, 20].

In this paper we present a statistical text summarizer based on Machine Learning (ML) and the text-span extraction paradigm. Our approach allows both generic and query-based summaries. However for evaluation purposes, we present here results for a generic SDS. For a given document, the system provides a set of unordered extracts which are supposed to be the most relevant to its topics. Previous work on the application of machine learning techniques for SDS used the classification framework. Such approaches usually train a classifier, using a training set of documents and their associated summaries, to distinguish between summary and non-summary sentences [13, 26, 5, 1]. After training, these systems operate on unlabeled text by ranking sentences of a new document according to the output of the classifier. The classifier is learned by comparing its output to a desired output reflecting a global class information. Under this framework one assumes that all sentences from different documents are comparable with respect to this class information. This hypothesis holds for scientific articles [13, 26] but for a large variety of collections, documents are heterogeneous and their summaries depend much more on the content of their texts than on a global class information.

We explore a new ML approach for SDS based on *ranking*. The latter has been successfully used in other domain-specific tasks such as Named-entity extraction [4] or metasearch [7]. We also believe that this framework is more adapted to SDS than the usual classification approach. It allows to learn a system with a weaker hypothesis on document sentences than the one assumed in classification. This makes the approach more efficient on other types of collections than scientific articles. The aim here is to combine automatically different features, giving each a relative ranking of sentences in a document, in order to achieve a high accurate ranking for summary sentences. For this combination, we propose generic and word cluster queries. The latter are new for SDS. To this aim, we group words occuring in the same sentences into a much smaller number of word-clusters than the initial vocabulary and use these clusters as features.

The paper is organized as follows, in section 2, we describe the features associated to each sentence, and in section 3 we show that ranking is more adapted to SDS than the

classification framework. We then detail the proposed ranking method for SDS in section 4 and show empirically in section 5 that the latter approach outperforms a state-of-art classifier on SUMMAC Cmp_lg and WIPO datasets.

2 Features for Text Summarization

A generic summary of a document has to reflect its key points. We need here statistical features which give different information about the relevance of sentences for the summary. If these features are sufficiently relevant for the SDS task, one can expect that they assign high scores to summary sentences in a document but rank them differently. We argue that there exists an optimal combination of these features which gives better results than the performance of the best feature. These Features constitute the input of the ML algorithms we developed here.

[18] defined different sentence features he considered important for a generic summary and grouped them into seven categories: *Indicator phrases* (such as cue-words or acronyms), *Frequency and title keywords*, *location* as well as *sentence length cutt-off* heuristics and the number of *semantic links* between a sentence and its neighbours. These features have partially or completely been used in the state of the art since then [13, 15, 9].

We also build from his work taking the *Indicator phrases* and *title keywords* features. As a feature gives a score to sentences in a document, we represent a ranking feature as a couple $(q, sim(q, s))$ where q is a generic query and $sim(q, s)$ is the similarity between q and a sentence s. In the following, we present different generic queries and similarity measures we used in this work.

2.1 Generic Queries

We start from two baseline generic queries constituted of the most frequent terms in the collection, **MFT** and no-stop words in the title of a document **title keyword**. These queries represent two sources of evidence we use to find relevant sentences in a document. Since **title keywords** may be very short, we have employed query-expansion techniques such as Local Context Analysis (LCA) [28] or thesaurus expansion methods (i.e. WordNet [6]) as well as a learning based expansion technique.

Expansion via WordNet and LCA: From the **title keyword** query, we formed two other queries, reflecting local links between the title keywords and other words in the corresponding document:

- **title keyword and LCA**, constituted by keywords in the title of a document and the most frequent words in most similar sentences to the **title keyword** query according to the cosine measure.
- **title keyword and most frequent terms**, constituted by high frequency document words and the keywords in the title of a document,

We also obtained an expanded query from the title keywords of a document and their first order synonyms using WordNet, **title keyword and WordNet**.

We propose next an unsupervised learning approach to expand the **title keyword** query. Such a technique allows one to find global links between words in a title of a document and words in the document collection.

Expansion with Word-Clusters: We first form different word-clusters based on words co-occurring in sentences of all documents in the corpus [3]. For discovering these word-clusters, each word w in the vocabulary V is first characterized as an p-dimensional vector $\mathbf{w} =< n(w,i) >_{i\in\{1,\ldots,p\}}$ representing the number of occurrences of w in each sentence i. Under this representation, word clustering is then performed using the `Naive-Bayes` clustering algorithm maximizing the Classification Maximum Likelihood criterion [23]. We have arbitrary fixed the number of clusters to be found to $\frac{|V|}{100}$.

From these clusters we obtained two other expanded queries by first adding to title keywords, words in their respective clusters, **title keyword and term-clusters**. And secondly by projecting each sentence of a document and the **title keyword** query in the space of these word-clusters, **Projected title keyword**. For the latter we characterize each sentence in a document and the **title keyword** query by a vector where each characteristic represents the number of occurrences of words from a cluster in that sentence or in the **title keyword** query. The characteristics in this representation are related to the degree of representation of each word-cluster in a given sentence or in the **title keyword** query.

2.2 Similarity Measures

Following [12], we use the *tf-idf* representation and compute the cosine similarity measure between sentence x and query q as :

$$Sim_1(q,s) = \frac{\sum_{w\in s,q} tf(w,q)tf(w,s)idf^2(w)}{\|w\|\|s\|}$$

Where, $tf(w,x)$ is the frequency of word w in x (q or s), $idf(w)$ is the inverse document frequency of word w and $\|x\| = \sqrt{\sum_{w\in x}(tf(w,x)idf(w))^2}$.

We also expected to reweight sentences containing acronyms e.g. HMM (Hidden Markov Models), NLP (Natural Language Processing), ... The resulting feature computes similarity between the **title keywords** and sentences using the same similarity measure than Sim_1 except that acronyms are given a higher weight. The resulting similarity measure writes

$$Sim_2(q,s) = \frac{\sum_{w\in s,q} tf(w,q)tf^*(w,s)idf^2(w)}{\|w\|\|s\|}$$

Hence, we have counted as twice the term frequency of acronyms e.g. $tf^*(w,s) = 2 * tf(w,s)$ if w is an acronym. In our experiments, acronyms are extracted using the *Acronym Finding Program* described in [24].

We have conducted experiments on scientific articles. For these documents, sentences containing any of a list of fixed phrases like "in this paper", "in conclusion", ... are more likely to be in summaries. We counted as twice the similarity of sentences containing such cue-words : $Sim_3(q,s) = 2Sim_1(q,s)$ if s contains cue-terms and $Sim_3(q,s) = Sim_1(q,s)$ if s does not contain cue-terms.

Table 1. Ranking features

#	Ranking features	(q, sim)
1	`Title`	(**title keywords**, Sim_1)
2	`Title+LCA`	(**title keywords and LCA**, Sim_1)
3	`Title+WN`	(**title keywords and WordNet**, Sim_1)
4	`Title+MFT`	(**title keywords and most frequent terms**, Sim_1)
5	`Title+Term-clusters`	(**title keywords and term-clusters**, Sim_1)
6	`Title+Acronyms`	(**title keywords**, Sim_2)
7	`Title+Cue words`	(**title keywords**, Sim_3)
8	`CommonTerms`	(**title keywords**, Sim_4)
9	`SumOfIdfs`	(**title keywords**, Sim_5)
10	`Projected title`	(**Projected title keywords**,Sim_6)
11	`GenericMFT`	(**MFT**, Sim_1)

Based on the first similarity measure we have also introduced three other similarities; $Sim_4(q, s) = \sum_{w \in s,q} 1$ computing the number of common words in the query q and a sentence s, $Sim_5(q, s) = \sum_{w \in s,q} idf(w)$ the sum of idf's of words in common in q and s and $Sim_6(q, s) = q.s$ the dot product between q and s.

The ranking features we considered are then constituted of 11 couples, (query, similarity), shown in table 1.

3 Ranking for Text Summarization

In order to combine sentence features, ML approaches for SDS adopt a classification framework, either using the Naive Bayes model [13] or a logistic regression classifier [1]. The motivation for such approaches is that a classification training error of 0 implies that scores assigned to relevant/irrelevant sentences from a classifier are all greater/lower than a constant c, resulting in an appropriate rankings of sentences.

However, on real life applications, this classification error is never zero. In this case, for a given document, we cannot predict about the ranking of a misclassified sentence relatively to the other ones. The reason is that the classification error is computed by comparing sentence scores with respect to a constant, and not relatively to each other. It can then happen that a misclassified irrelevant sentence gets higher score than relevant ones. In other terms, minimizing the classification error does not necessary leads to the optimization of the ranks of relevant sentences in the same document.

We believe that algorithms relying on the ML *ranking* framework will be more effective in practice for the SDS task. In this case, instead of classifying sentences as relevant/irrelevant, a ranking algorithm classifies pairs of sentences. More specifically, it considers the pair of sentences $(\mathbf{s}, \mathbf{s}')$ coming from a same document, such that just one of the two sentences is relevant. The goal is then to learn a scoring function H from the following assumption: a pair is correctly classified if and only if the score of the relevant sentence is greater than the score of the irrelevant one. The error on the pairs of sentences, called the *Ranking loss* of H [7], is equal to:

$$Rloss(\mathcal{D}, H) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|\mathcal{S}_1^d||\mathcal{S}_{-1}^d|} \sum_{\mathbf{s} \in \mathcal{S}_1^d} \sum_{\mathbf{s}' \in \mathcal{S}_{-1}^d} [[H(\mathbf{s}') \geq H(\mathbf{s})]] \quad (1)$$

where $\mathcal{D}$ is the training document collection, $\mathcal{S}_1^d$ the set of relevant sentences for document d, and $\mathcal{S}_{-1}^d$ the set of the irrelevant ones of the same document and $[[\pi \geq 0]]$ is equal to 1 if $\pi \geq 0$ holds and 0 in the contrary case.

It is straightforward that minimizing *Ranking loss* is equivalent to minimize the number of irrelevant sentences scored higher than the relevant ones of the same document. *Ranking loss* results then in a direct optimization of the ranks of the relevant sentences. This fact motivates the use of a ranking algorithm instead of a classification algorithm for the SDS task.

3.1 Logistic Regression Classifier for SDS

The logistic regression has already been used for SDS [1]. It has shown good empirical results in terms of Precision/Recall for the combination of features. We will see in the next section that an efficient ranking algorithm can be naturally derived from the logistic regression.

As input of the logistic classifier, we represent each sentence $\mathbf{s}$ by a vector of scores $(s_1, ..., s_n)$, where the score s_i is given by the feature i (Table 1).

The logistic classifier makes the following assumption on the form of the posterior probability of the class *relevant* given a sentence $\mathbf{s}$:

$$P(relevant|\mathbf{s}) = \frac{1}{1 + e^{-2\sum_{i=1}^{n} \lambda_i s_i}}$$

And learns the parameters $\Lambda = (\lambda_1, ..., \lambda_n)$ by maximizing the binomial log-likelihood [8], which writes:

$$\mathcal{L}(\mathcal{D}; \Lambda) = -\frac{1}{2} \sum_{y=-1,1} \frac{1}{|\mathcal{S}^y|} \sum_{\mathbf{s} \in \mathcal{S}^y} \log(1 + e^{-2y\sum_{i=1}^{n} \lambda_i s_i}) \quad (2)$$

where $\mathcal{D}$ is the set of training documents, and $\mathcal{S}^{-1}$ and $\mathcal{S}^1$ are respectively the set of relevant and irrelevant sentences in the training set and $y \in \{-1, 1\}$ (1 represents the class of the relevant sentences).

3.2 Adaptation to Ranking for SDS

There exist several ranking algorithms in the ML literature, based on the perceptron [4, 21] or *AdaBoost* - called *RankBoost* [7]. For the SDS task, as the total number of sentences in the collection may be very high we need a simple and efficient ranking algorithm. Perceptron-based ranking algorithms would lead to quadratic complexity in the number of examples, while the *RankBoost* algorithm in its standard setting does not search a linear combination of the input features. For the sake of simplicity, we compare in this paper a linear classifier with a linear ranker - called `LinearRank` in the following - which combines both efficiency (complexity linear in the number of examples) and simplicity.

We represent the pair $(\mathbf{s}, \mathbf{s}')$ by the difference of their representative vectors, $(s_1 - s'_1, ..., s_n - s'_n)$. We want to learn a scoring function $H(s) = \sum_{i=1}^{n} \theta_i s_i$, for any sentence s in the collection. The *Ranking loss* (1) can be written as the following:

$$Rloss(\mathcal{D}, H) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|\mathcal{S}_1^d||\mathcal{S}_{-1}^d|} \sum_{\mathbf{s} \in \mathcal{S}_1^d} \sum_{\mathbf{s}' \in \mathcal{S}_{-1}^d} [[\sum_{i=1}^{n} \theta_i (s'_i - s_i) \geq 0]]$$

This expression is a standard linear classification error, on the pairs of sentences represented by the difference of the sentence vectors. We can then adapt any linear classification algorithm to ranking (logistic regression in our case) in order to optimize the previous criterion.

The logistic assumption, adapted to ranking, becomes:

$$P(1|\, \mathbf{s}, \mathbf{s}') = \frac{1}{1 + e^{-2 \sum_{i=1}^{n} \theta_i (s_i - s'_i)}}$$

where $\mathbf{s}$ is a relevant sentence for a given document, and $\mathbf{s}'$ an irrelevant sentence for the same document. $P(1|\, \mathbf{s}, \mathbf{s}')$ denotes the posterior probability that the considered pair is well classified.

The parameters $\Theta = (\theta_1, ..., \theta_n)$ are learned by maximizing the corresponding binomial log-likelihood:

$$\mathcal{L}(\mathcal{D}; \Theta) = -\frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|} \sum_{(\mathbf{s}, \mathbf{s}') \in \mathcal{S}_d^1 \times \mathcal{S}_d^{-1}} \log(1 + e^{-2 \sum_{i=1}^{n} \theta_i (s_i - s'_i)}) \qquad (3)$$

where $\mathcal{D}$ is the set of training documents, and, for $d \in \mathcal{D}$, $\mathcal{S}_d^1$ is the set of relevant sentences in d and $\mathcal{S}_d^{-1}$ the set of irrelevant ones.

[8] have shown that the optimization of (3) leads to the same parameters as minimizing the exponential loss[1]:

$$\mathbf{ELoss}(\mathcal{D}; \Theta) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|} \sum_{(\mathbf{s}, \mathbf{s}') \in \mathcal{S}_d^1 \times \mathcal{S}_d^{-1}} e^{\sum_{i=1}^{n} \theta_i (s'_i - s_i)} \qquad (4)$$

This latter function is convex, so standard optimization algorithms can be used to minimize it. In our case, we used an iterative scaling algorithm to learn the parameters, which is an adaptation for ranking of an algorithm developed for classification described in [11]. The interesting property of the exponential loss is that in our case, it can be computed in time linear in the number of examples, simply by rewriting (4) as follows:

$$\mathbf{ELoss}(\mathcal{D}; \Theta) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|} \Big(\sum_{\mathbf{s}' \in \mathcal{S}_d^{-1}} e^{\sum_{i=1}^{n} \theta_i s'_i} \Big) \Big(\sum_{\mathbf{s} \in \mathcal{S}_d^1} e^{-\sum_{i=1}^{n} \theta_i s_i} \Big) \qquad (5)$$

[1] It is interesting to note that this exponential loss is the one minimized by the *RankBoost* algorithm [7] and is intuitively related to the ranking problem, because the following inequality holds: $\mathbf{ELoss}(\mathcal{D}; \Theta) \geq \mathbf{Rloss}(\mathcal{D}; \Theta)$. No such inequality can be found between the ranking loss and the binomial likelihood.

On the opposite, the computation of the maximum likelihood of equation (3) requires to consider all the pairs of sentences, and leads to a complexity quadratic in the number of examples. Thus, although ranking algorithms consider the pairs of examples, in the special case of SDS, the proposed algorithm is of complexity linear in the number of examples through the use of the exponential loss.

In order to compare equitably between classification and ranking for SDS, we employed in both cases the same logistic model but trained it differently depending on the framework in use. Hence, we trained the model in a classification framework by maximizing the binomial likelihood criterion (2). While the model parameters are learned by minimizing the **ELoss** criterion (5) in the case of ranking.

4 Experiments

4.1 Data Sets: Properties

A good extractive summarizer has to find the relevant information for which the user is looking as well as to eliminate the irrelevant one. It is then crucial to evaluate the system on the way it is able to identify how well it can extract the pieces of articles that are relevant to a user. To this end we used two datasets from the `SUMMAC cmp_lg` evaluations sets [22] and the `WIPO` collection [27].

The `SUMMAC` corpus is constituted of 183 articles. Documents in this collection are scientific papers which appeared in ACL sponsored conferences. The collection has been marked up in xml by converting automatically the latex version of the papers to xml. The second data set, `WIPO`, is an automated categorization collection which contains over 75000 patent documents in English. Documents in this collection are also marked up in xml. In our experiments, we have chosen 1000 documents in random from this corpus. In both datasets markup include tags covering information such as title, authors or inventors, etc., as well as basic structure such as abstract, body, sections, lists, etc.

In order to find the relevant information in documents, we have used the text-span alignment method descried by [16] to generate extract-based summaries from the abstract of each document. In this method, summaries required for training and evaluation are automatically generated as follows: from a pool of all sentences in a document, Marcu's algorithm discard those which removal increases the similarity between the rest of the pool and the abstract. And this until that any removal decreases the similarity measure.

For learning systems, an advantage of the Marcu's algorithm is that, in the case of huge datasets, *gold* summaries are not available. The human extraction of such reference summaries is infeasible. Moreover in [16] Marcu has proven empirically that the performance of his alignment algorithm is close to that of humans by means.

Contrarily to the `SUMMAC` corpus, `WIPO` collection is constituted from heterogeneous documents in which a relevant sentence from a document may have a completely different feature representation than another relevant sentence from another document. It is then interesting to see the behaviour of the ranking and the classification algorithms in such a corpus where relevant sentences in different documents are mapped into different parts of the feature space.

Table 2. Data Set properties

Data set comparison		
source	SUMMAC	WIPO
Number of docs	173(183)	854(1000)
Average # of sent. per doc.	156.57	179.63
Maximum # of sent. per doc.	928	1507
Minimum # of sent. per doc.	15	21
Number of doc. in (training-test) sets	73-100	360-494
Average # of words per sent.	11.73	14.22
Size of the vocabulary	15621	56856
Summary as % of doc. length	10	10
Average summary size (in # of sent)	11.44	6.07
Maximum # of sent. per summary	27	19
Minimum # of sent. per summary	3	2

Documents are tokenized by removing *xml* tags as well as words in a stop list and sentence boundaries within each document are found using the morpho-syntactic tree-tagger program [25]. In each data collection, low collection frequency words (occurring in less than two documents) are also removed. A compression ratio must be specified or computed for extractive summaries. For both datasets we followed the SUMMAC evaluation by using a 10% compression ratio [22].

From each dataset, we removed documents having summaries (found by Marcu's algorithm) composed of 1 sentence arguing that a sentence is not sufficient to summarize a scientific or a patent article. From the WIPO data collection, we have also removed documents having less than 2 words in their title. In total we have removed respectively 10 documents from SUMMAC and 146 from WIPO collections. Table 2 gives the characteristics of both datasets.

4.2 Results

Evaluation issues of summarization systems have been the object of several attempts, many of them being carried within the tipster program and the summac competition. This is a complex issue and many different aspects have to be considered simultaneously in order to evaluate and compare different summaries. For the extraction task we are dealing here, things are a bit easier. We compared the extract of a system with the desired summary at 10% compression ratio and used the following Precision and Recall measures to plot the curves:

$$\text{Precision} = \frac{\text{\# of sentences extracted by the system which are in the target summaries}}{\text{total \# of sentences extracted by the system}}$$

$$\text{Recall} = \frac{\text{\# of sentences extracted by the system which are in the target summaries}}{\text{total \# of sentences in the target summaries}}$$

For each dataset, we ran the ranking algorithm (LinearRank), the logistic classifier and statistical features giving scores to sentences. First we sought to show the ability of query expansion techniques (without learning effects). Expansion using LCA

and WordNet thesaurus were found to be effective for summarization; we went further by introducing the expansion based on word-clusters. The benefits of query expansion for summarization consisted of comparing the **Title** feature to **Title+LCA**, **Title+WN**, **Title+MFT**, **Projected Title** and **Title+Term-clusters** features. The results of this experiment are shown in Figure 1.

These results show that the three best features are **Title+LCA**, **Projected Title** and **Title+Term-clusters**. It comes that local and global query expansions improve the performance of the baseline **title keywords** query for SDS. However we note that the performance of LCA varies between the two datasets: on the SUMMAC corpus, **Title+LCA** has 70% precision for 50% recall while the **Title** feature gives a 40% precision. On the WIPO corpus the difference between the precisions of these two features is reduced to approximately 6% for the same recall. This may be due to the fact that there are fewer relevant sentences in documents from the `WIPO` dataset (see table 2). Thus, it is possible that more irrelevant sentences are used in the computation of co-occurences of words for LCA. The difference between the two features **Title+Term-clusters** and **Projected Title** is that the first one does not take into account all the words from the word-clusters, while the second one considers sentences and the title in the cluster space. This consideration leads to a different computation of idf weights for the second query, which is highly affected by the number of clusters. In our experiments, **Title+Term-clusters** performs better in both datasets. This may be due to the fact that the clusters contain too much irrelevant words, which makes the feature **Projected Title** give high scores to some irrelevant sentences. Consequently, some future work is needed to study the effect of the optimal number of word-clusters and to fully understand the effect of representing sentences in the cluster space instead of adding words into the query.

The performance of the learning algorithms are plotted in figure 2. In both datasets, these algorithms perform better than statistical features while the ranking algorithm outperforms the logistic classifier. This means basically that the two learning frameworks lead to a good combination of statistical features, but the ranking framework is more adapted to SDS than the classification framework.

On the SUMMAC corpus, the difference in terms of precision between classification and ranking vary from 2% to 5% at different levels of recall. On the WIPO corpus, it varies between 5% and 9%. The difference between the classification algorithm and the best feature varies from 3% to 9% on the SUMMAC corpus, and from 0% to 5% on the WIPO corpus at different levels of recall. This shows that the performance of the combination of features found by the classifier, compared to the best feature, vary a lot depending on the corpus, while the ranking algorithm finds an accurate combination of features on both datasets.

An analysis of the weights given by both learning algorithms to different features shows that the most important features in the combination are **Title**, **Title+LCA**, **Title+Term-clusters**, **Projected Title** and **generic query**. It comes that learning algorithms give importance to features upon two criteria: firstly, their ability to give high scores to relevant sentences and, secondly, their independence with other features. Thus, the **generic query** which gives the worst performance in our experiments, is given a higher weight by the ranking algorithm than features such as **Title+WN** or **Title+Cue Words** which are highly correlated to the **Title** feature.

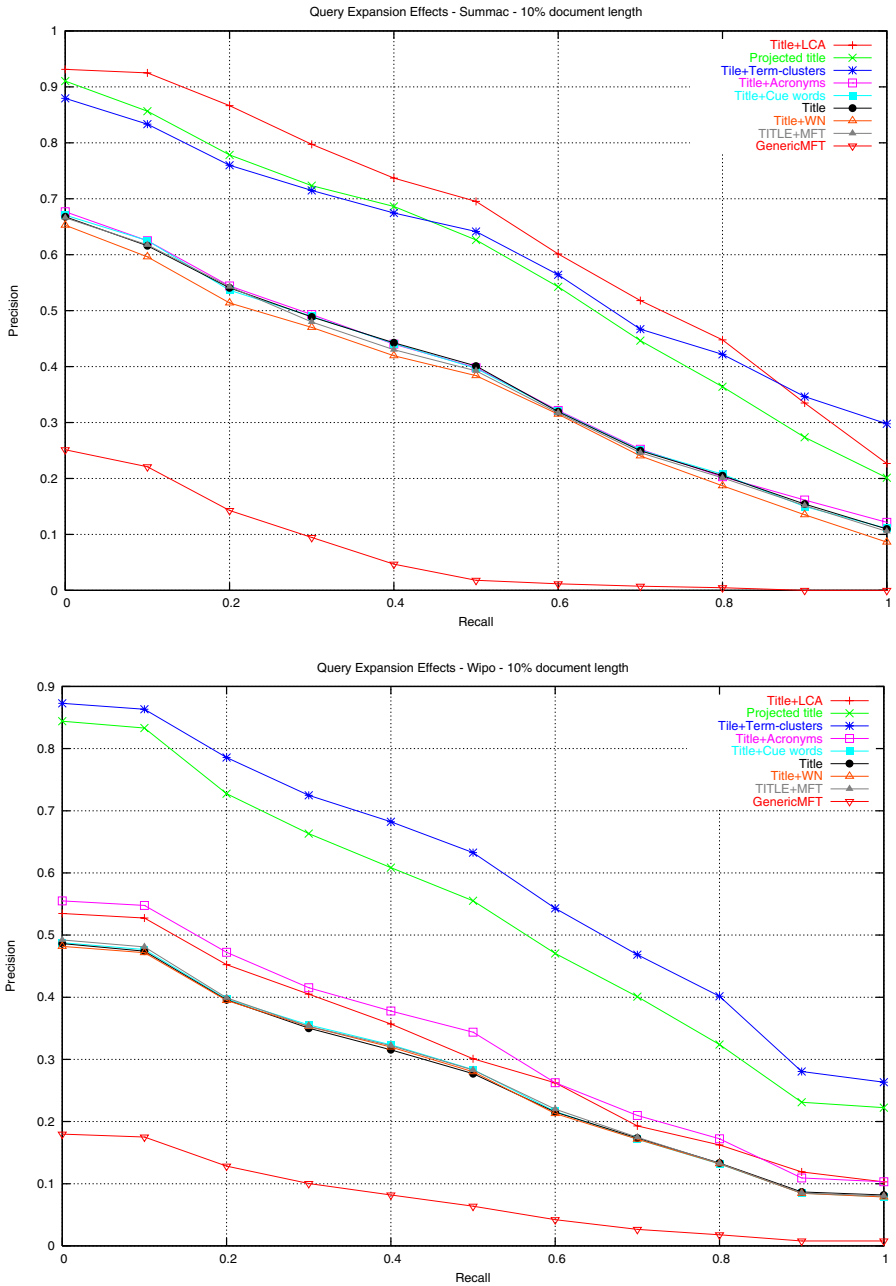

Fig. 1. Query expansion effects on SUMMAC (top) and WIPO (down) datasets

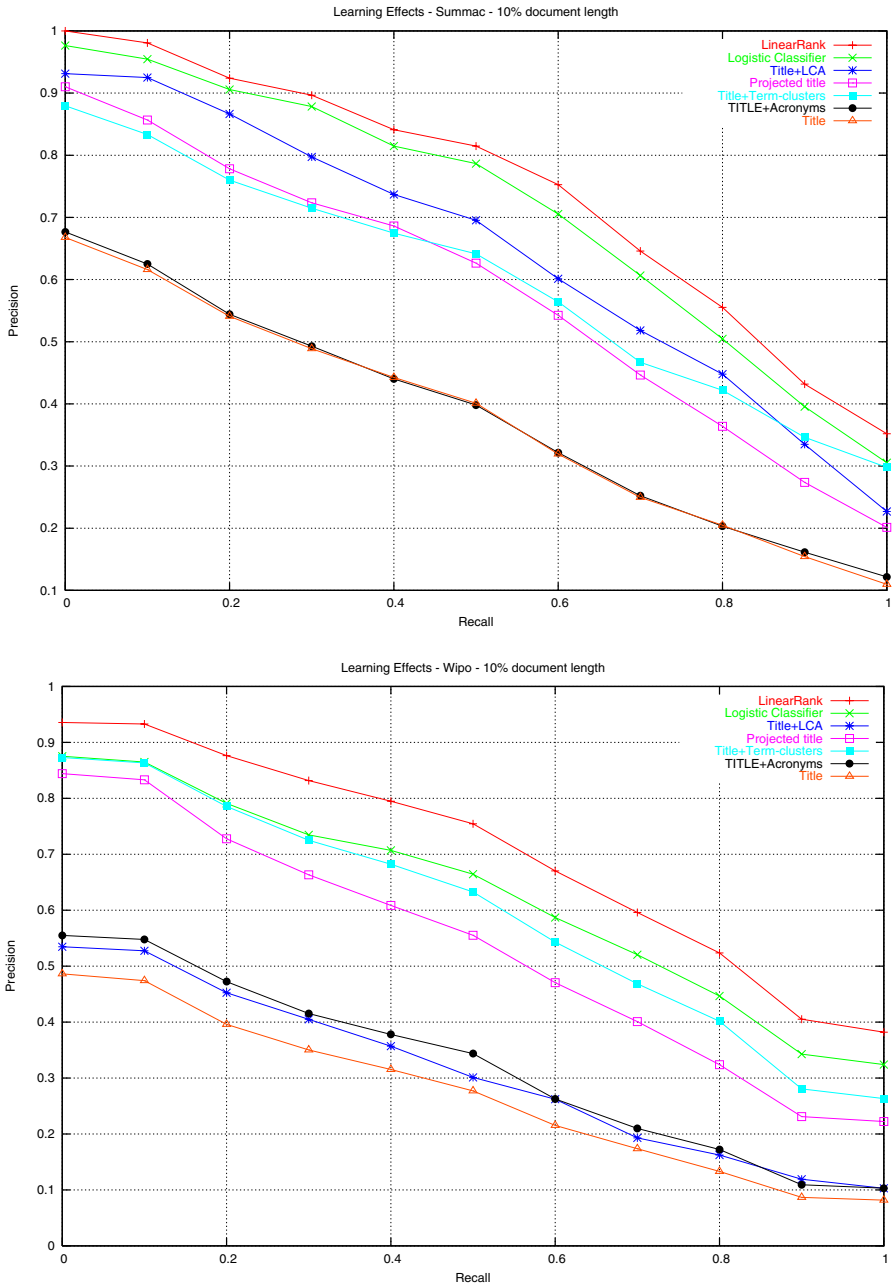

Fig. 2. Learning effects on SUMMAC (top) and WIPO (down) datasets

This is consistent with observations in metasearch which show that in order to have a highly accurate combination, one need to have independent features [2]. Moreover, it confirms the interest of the words clusters for text summarization, since they provide information independent from the other performing features like **Title+LCA**.

5 Discussion

The empirical results lead to two interesting remarks. Firstly, the ranking algorithm outperforms the classification algorithm on both datasets. Secondly, the difference of performance between the two algorithms depends on the nature of the collection.

We can explain the difference of performance between classification and ranking by the difference of their optimization criterion, but a deeper analysis is needed to fully understand why the difference of performance strongly depends on the data set.

For the sake of simplicity, we restrict our interpretation on linear classification and ranking algorithms. The hypothesis on the dataset made by a linear ranker is that relevant and irrelevant sentences of a given document are separated in the feature space by a hyperplane. For all documents in a data collection, the underlying hyperplanes which separate relevant sentences from irrelevant ones are all parallel. On the other hand, the hypothesis made by a linear classifier is that there exists a unique hyperplane separating all relevant sentences from all irrelevant ones. This latter hypothesis is a particular case of the linear ranking hypothesis, where, among documents, hyperplanes are not only parallel, but equal.

This remark enables us to explain that the difference of performance between classification and ranking depends on the document collection. On homogeneous datasets, the separating hyperplanes will be approximately the same for all documents, resulting in a small difference of performance between ranking and classification (which is probably the case for the SUMMAC corpus). On more heterogeneous datasets, like the WIPO corpus, the separating hyperplanes will be more distant in the feature space. In this case, the dataset follows no longer the working assumption of the linear classifier, which consequently finds a suboptimal separating hyperplane, leading to more important differences between classification and ranking.

6 Conclusion

In conclusion, we have presented new features for text summarization, and proposed the use of ranking algorithms to combine these features.

The features introduced are based on word clusters, which group words co-occurring in the same sentences. These clusters can be used to provide words for query expansion, or to enrich the representation of the sentences. In all cases, they show promising performance in terms of precision/recall, but future work is needed to fully understand the difference between the two techniques as well as studying the effect of the number of clusters and their size on the performance of the features. Moreover, they bring additional and independent information to standard features used in SDS, and are therefore of great interest in the case where we want to build an accurate combination of features.

To the best of our knowledge, this paper is the first one to propose the use of a ML ranking algorithm for SDS. We have shown empirically that ranking algorithms outperform classification algorithms. Ranking algorithms have a weaker working hypothesis than classification algorithms, and seem more appropriate to the SDS, although the difference of performance between the two depends on the dataset we are studying. However, important gains can be expected on specific datasets, while it is probable that classification algorithms can do worse.

This understanding of the behavior of ranking algorithms can lead to its use on other tasks of passage level extraction, where the optimization criterion as well as the working hypothesis of ranking algorithms may be more suited than classification algorithms.

Acknowledgments

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors views.

References

1. Amini M.-R., Gallinari P.: The Use of unlabeled data to improve supervised learning for text summarization. Proceedings of the 25^{th} ACM SIGIR, 105–112, (2002).
2. Aslam, J.A., Montague, M.: Models for metasearch. In Proceedings of the 24^{th} annual international ACM SIGIR conference on Research and development in information retrieval, (2001)
3. Caillet M., Pessiot J.-F., Amini M.-R., Gallinari P.: Unsupervised Learning with Term Clustering for Thematic Segmentation of Texts. Proceedings of RIAO, (2004).
4. Collins, M. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)
5. Chuang W.T., Yang J.: Extracting sentence segments for text summarization: a machine learning approach. Proceedings of the 23^{th} ACM SIGIR, 152–159, (2000).
6. Fellbaum C.: WordNet, an Electronic Lexical Database. MIT Press, Cambridge MA (1998)
7. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research, **4** (2003) 933–969
8. Friedman J., Hastie T., Tibshirani R.: Additive Logistic Regression: a Statistical View of Boosting. Technical Report Stanford University, (1998).
9. Goldstein J., Kantrowitz M., Mittal V., Carbonell J.: Summarizing Text Documents: Sentence Selection and Evaluation Metrics. Proceedings of the 22^{th} ACM SIGIR, 121–127, (1999).
10. Jing H.: Summary generation through intelligent cutting and pasting of the input document. Technical Report Columbia University, (1998).
11. Lebanon, G., Lafferty J.: Boosting and maximum likelihood for exponential models. Technical Report CMU-CS-01-144, School of Computer Science, CMU (2001).
12. Knaus D., Mittendorf E., Shauble P., Sheridan P.: Highlighting Relevant Passages for Users of the Interactive SPIDER Retrieval System. In TREC-4 Proceedings (1994).
13. Kupiec J., Pederson J., Chen F.A.: Trainable Document Summarizer. Proceedings of the 18^{th} ACM SIGIR, 68–73, (1995).
14. Luhn P.H.: Automatic creation of litterature abstracts. IBM Journal, pp. 159–165 (1958).

15. Mani I., Bloedorn E.: Machine Learning of Generic and User-Focused Summarization. Proceedings of hte Fifteenth National Conferences on AI pp. 821–826 (1998).
16. Marcu D.: The Automatic Construction of Large-Scale corpora for Summarization Research. Proceedings of the 22^{th} ACM SIGIR, (1999).
17. Mitra M., Singhal A., Buckley C.: Automatic Text Summarization by Paragraph Extraction. Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, pp. 31–36 (1997).
18. Paice C.D., Jones P.A.: The identification of important concepts in highly structured technical papers. Proceedings of the 16^{th} ACM SIGIR, 69–78, (1993).
19. Sparck-Jones K.: Discourse modeling for automatic summarizing. Technical Report 29D, Computer Laboratory, university of Cambridge, (1993).
20. Strzalkowski T., Wang J., Wise B.: A Robust practical text summarization system. Proceedings of hte Fifteenth National Conferences on AI pp. 26–30 (1998).
21. Shen, L., Joshi, A.K.: Ranking and Reranking with Perceptron. Machine Learning, Special Issue on Learning in Speech and Language Technologies (2004)
22. http://www.itl.nist.gov/iaui/894.02/related_projects/tipster_summac/cmp_lg.html
23. Symons M.J.: Clustering Criteria and Multivariate Normal Mixture. Biometrics Vol. 37. 35–43 (1981).
24. Taghva K., Gilbreth J.: Recognizing acronyms and their definitions. IJDAR Vol. 1. 191–198 (1999).
25. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html
26. Teufel S., Moens M.: Sentence Extraction as a Classification Task. Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, pp. 58–65 (1997).
27. http://www.wipo.int/ibis/datasets/index.html
28. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval (1996).
29. Zechner K.: Fast Generation of Abstracts from General Domain Text Corpora by Extracting Relevant Sentences. COLING, 986–989, (1996).