Learning-Based Summarisation of XML Documents

Massih R. Amini[†] Anastasios Tombros[*] Nicolas Usunier[†] Mounia Lalmas[*]

[†]`{name}@poleia.lip6.fr`
[†]University Pierre and Marie Curie
8, rue du capitaine Scott
75015, Paris
France

[*]`{first_name}@dcs.qmul.ac.uk`
[*]Queen Mary, University of London
Department of Computer Science
London E1 4NS
United Kingdom

Abstract. Documents formatted in eXtensible Markup Language (XML) are available in collections of various document types. In this paper, we present an approach for the summarisation of XML documents. The novelty of this approach lies in that it is based on features not only from the content of documents, but also from their logical structure. We follow a machine learning, sentence extraction-based summarisation technique. To find which features are more effective for producing summaries, this approach views sentence extraction as an ordering task. We evaluated our summarisation model using the INEX and SUMMAC datasets. The results demonstrate that the inclusion of features from the logical structure of documents increases the effectiveness of the summariser, and that the learnable system is also effective and well-suited to the task of summarisation in the context of XML documents. Our approach is generic, and is therefore applicable, apart from entire documents, to elements of varying granularity within the XML tree. We view these results as a step towards the intelligent summarisation of XML documents.

1 Introduction

With the growing availability of on-line text resources, it has become necessary to provide users with systems that obtain answers to queries in a manner which is both efficient and effective. In various information retrieval (IR) tasks, single document text summarisation (SDS) systems are designed to help users to quickly find the needed information [19, 24]. For example, SDS can be coupled with conventional search engines and help users to evaluate the relevance of documents [34] for providing answers to their queries.

The original problem of summarisation requires the ability to understand and synthesise a document in order to generate its abstract. However, different attempts to produce human quality summaries have shown that this process of abstraction is highly complex, since it needs to borrow elements from fields such as linguistics, discourse understanding and language generation [23, 16]. Instead, most studies consider the task of text summarisation as the extraction of text spans (typically sentences) from the original document; scores are assigned to text units and the best-scoring spans are presented in the summary. These approaches transform the problem of abstraction into a simpler problem of *ranking* spans from an original text according to their relevance to be part of the document summary. This kind of summarisation is related to the task of document retrieval, where the goal is to rank documents from a text collection with respect to a given query in order to retrieve the best matches. Although such an extractive approach does not perform an in-depth analysis of the source text, it can produce summaries that have proven to be effective [19, 24, 34].

To compute sentence[1] scores, most previous studies adopt a linear weighting model which combines statistical or linguistic features characterising each sentence in a text [21]. In many systems, the set of feature weights are tuned manually; this may not be tractable in practice, as the importance of different features can vary for different text genres [14]. Machine Learning (ML) approaches within the classification framework, have shown to be a promising way to combine automatically sentence features [17, 32, 5, 2]. In such approaches, a classifier is trained to distinguish between two classes of sentences: summary and non-summary ones. The classifier is learnt by comparing its output to a desired output reflecting global class information. This framework is limited in that it makes the assumption that all sentences from different documents are comparable with respect to this class information.

Here we explore a ML approach for SDS based on ranking. The main rationale of this approach is to learn how to best combine sentence features such that within each document, summary sentences get higher scores than non-summary ones. This ordering criterion corresponds exactly to what the learnt function is used for, i.e. ordering sentences. Statistical features that we consider in this work, are partly from the state-of-art, and they include cue-phrases and positional indicators [21, 9], and title-keyword similarity [9]. In addition, we propose a new contextual approach based on topic identification to extract meaningful features from sentences.

In this paper, we apply the ML approach for summarisation to XML documents. The XML format is becoming increasingly popular [26], and this has caused a considerable interest in the content-based retrieval of XML documents, mainly through the INEX initiative [13]. In XML retrieval, document components, rather than entire documents, are retrieved. As the number of XML components is typically large (much larger than that of documents), it is essential to provide users of XML IR systems with overviews of the contents of the retrieved elements. The element summaries can then be used by searchers in an interactive environment. In traditional (i.e. non XML) interactive information retrieval, a summary is usually associated with each document; in interactive XML retrieval, a summary can be associated with each retrieved XML component. Because of the nature of XML documents, users can also browse within the XML document containing that element. One method to facilitate browsing, is to display the logical structure of the document containing the retrieved elements (e.g. in a Table of Contents format). In this way, summaries can also be associated with the other elements forming the document, in addition to the retrieved elements themselves [30]. The choice of the ”meaningful” granularity of elements to be summarised is also currently being investigated [31], as some retrieved elements may simply be too short to be summarised. The summarisation of XML documents is also beginning to draw attention from researchers [1, 20, 26, 30].

[1] In our experiments we have considered sentences for extractive summarisation, so from now on, we will refer to sentences as the basic text-units to be extracted.

A major aim of this paper is to investigate the effectiveness of an XML summarisation approach by combining structural and content features to extract sentences for summaries. More specifically, a further novel feature of our work is that we make use of the logical structure of documents to enhance sentence characterisation. In XML documents, a tree-like structure, which corresponds to the logical structure of the source document, is encoded. For example, an article can be seen as the root of the tree, and sections, subsections and paragraphs can be arranged in branches and leaves of the tree. We select a number of features from this logical structure, and learn what features are best predictors of ”summary-worthy” sentences.

The contributions of this work are therefore twofold: first, we propose and justify the effectiveness of a ranking algorithm, instead of the mostly used classification error criterion in ML approaches for SDS, and second, we investigate the summarisation of XML documents by taking into account features relating both to the content and the logical structure of the documents. The ultimate aim of our approach is to generate summaries for components of XML documents at any level in the logical structure hierarchy. Since at present the evaluation of such summaries is hard (due to the lack of appropriate resources), we consider an XML article to be an XML element, and we use its content and structure to learn how we can best summarise it. Our approach is sufficiently generic to be applied to a component at any level of the logical structure of an XML document.

In the remainder of the paper, we first discuss, in section 2, related work on ML approaches based on the classification framework and outline our ML approach for summarisation. In section 3 we present the structural and content features that we used to represent sentences for this task. In Section 4 we outline our evaluation methodology. In section 5 we present the results of our evaluation using two datasets from the INitiative for the Evaluation of XML retrieval (INEX) [13] and the Computation and Language collection (cmp-lg) of TIPSTER SUMMAC [28]. Finally, in section 6 we discuss the outcomes of this study and we also draw some pointers for the continuation of this research.

2 Trainable text summarisers

The purpose of this section is to present evidence that, for SDS, a ranking framework is better suited for the learning of a scoring function than a classification framework. To this end, we define two trainable text summarisers learnt using a classification and a ranking criterion, and show upon the choice of these learning criteria why our proposition holds. In both cases, we aim to learn a scoring function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ which represents the best linear combination of sentence features according to the learning criterion in use under the supervised setting. We chose to use a simple linear combination of sentence features for two reasons. First, under the classification framework, it has been shown that simple linear classifiers like the Naive Bayes model [17], or a Support Vector Machine [15] perform as well as more complex non-linear classifiers [5]. Secondly, in order to compare fairly between the ranking and classification approaches we fix the class of the scoring function (linear in our case) and consider two different learning criteria developed under these two frameworks. The choice of the best ranking function class for SDS is beyond the scope of the paper.

In the following, we first present notations used in the rest of the paper and give a brief review of the classification framework for text summarisation, and then present the main motivation for using an alternative ML approach based on ordering criteria for this task.

2.1 Notations

We denote by $\mathcal{D}$ the collection of documents in the training set and assume that each document d in $\mathcal{D}$ is composed of a set of sentences[2], $d = (s^k)_{k\in\{1,\dots,|d|\}}$ where $|d|$ is the length of document d in terms of the number of sentences composing d. Each sentence $s = (s_i)_{i\in\{1,\dots,n\}}$ is characterised by a set of n structural and statistical features that we present in Section 3. Without loss of generality, we assume that every feature is a positive real value for any sentence. Under the supervised setting, we suppose that a binary relevance judgment vector $y = (y^k)$, $y^k \in \{-1, 1\}, 1 \leqslant k \leqslant |d|$ is associated to each document d; y^k indicates whether the sentence s^k in d belongs, or not, to the summary.

2.2 Text summarisation as a classification task

In this section, we present the classification framework for SDS which is the most used learning scheme for this task in literature. We first present a classification learning criterion related to the minimisation of the misclassification error, and then present a logistic classifier that we prove to be adequate for this optimisation.

Misclassification error rate The working principle of classification approaches to SDS is to associate class label 1 to summary (or *relevant*) sentences, and class label -1 to non-summary (or *irrelevant*) ones, and to use a learning algorithm to discover for each sentence s the best combination weights of its features $h(s)$, with the goal of minimising the error rate of the classifier (or its classification loss denoted by $L_{\mathcal{C}}$), that is, the expectation that a sentence is incorrectly classified by the output classifier.

$$L_{\mathcal{C}}(h) = \mathbb{E}\left([[yh(s) < 0]]\right) \tag{1}$$

where $[[pr]]$ is equal to 1 if predicate pr holds and 0 otherwise. The computation of this expected error rate depends on the probability distribution from which each pair (sentence, class) is supposed to be drawn identically and independently. In practice, since this distribution is unknown, the true error rate cannot be computed exactly and it is estimated over a labeled training set by the *empirical error rate* $\hat{L}_c$ given by

$$\hat{L}_{\mathcal{C}}(h, \mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}[[yh(s) < 0]] \tag{2}$$

[2] Recall that in extractive summarisation, the summary of a document is made of a subset of its sentences.

where $\mathcal{S}$ represents the set of all sentences appearing in $\mathcal{D}$. We notice here that sentences from different documents are comparable with respect to a global class information.

A direct optimisation of the empirical error rate (equation 2) is not tractable as this function is not differentiable. Schapire and Singer [25] motivate $e^{-yh(s)}$ as a differentiable upper bound to $[[yh(s) < 0]]$. This follows because for all x, $e^{-x} \geq [[x < 0]]$.

Figure 1 shows the graphs of these two misclassification error functions as well as the log-likelihood loss function introduced below with respect to yh; negative (positive) values of yh imply incorrect (correct) classification. The exponential and log-likelihood criteria are differentiable upper bounds of the misclassification error rate. These functions are also convex, so standard optimisation algorithms can be used to minimise them. Friedman et al. have shown in [12] that the function h minimising $\mathbb{E}(e^{-yh(s)})$ is a logistic classifier whose output estimates $p(y = 1|s)$, the posterior probability of the class *relevant* given a sentence s.

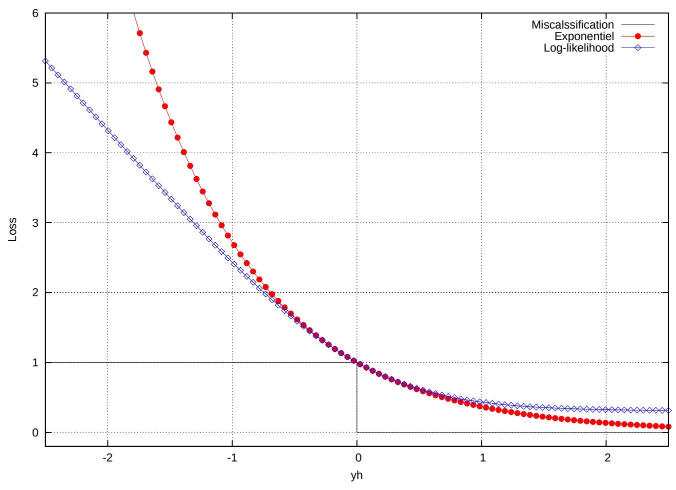


Fig. 1. Misclassification, exponential and log-likelihood loss functions with respect to yh.

In many ML approaches, the optimisation criterion to train a logistic classifier is the binomial log-likelihood function $-\mathbb{E}\log(1 + e^{-2yh(s)})$. The reason is that from a statistical point of view, $e^{-yh(s)}$ is not equal to the $\log$ of any probability mass function on ± 1 as it is the case for $-\log(1+e^{-2yh(s)})$. Nevertheless, Friedman et al. have shown that the optimisation of both criteria is effective and that the population minimisers of $-\mathbb{E}\log(1 + e^{-2yh(s)})$ and $\mathbb{E}(e^{-yh(s)})$ coincide [12].

For the ranking case, we will adopt a similar logistic model and show that the minimisation of the exponential loss has a real advantage over the log-binomial in terms of computational complexities (see Section 2.3).

Logistic model for classification For the classification case, we propose to learn the parameters $\Lambda = (\lambda_1, ..., \lambda_n)$ of the feature combination $h(s) = \sum_{i=1}^{n} \lambda_i s_i$ by training a logistic classifier whose output estimates $p(relevant \mid s) = \frac{1}{1+e^{-2h(s)}}$ in order to minimise the empirical exponential bound estimated on the training set:

$$L_{exp}^{c}(\mathcal{S}; \Lambda) = \frac{1}{|\mathcal{S}|} \sum_{y \in \{-1,1\}} \sum_{\mathbf{s} \in \mathcal{S}^y} e^{-y \sum_{i=1}^{n} \lambda_i s_i} \qquad (3)$$

where $\mathcal{S}^1$ and $\mathcal{S}^{-1}$ are respectively the set of relevant and irrelevant sentences in the training set $\mathcal{S}$ and $|\mathcal{S}|$ is the number of sentences in $\mathcal{S}$.

For the minimisation of L_{exp}^{c}, we employ an iterative scaling algorithm [7]. This procedure is shown in Algorithm 1. Starting from some arbitrary set of parameters $\Lambda = (\lambda_1, ..., \lambda_n)$, the algorithm finds iteratively a new set of parameters $\Lambda + \Delta = (\lambda_1 + \delta_1, ..., \lambda_n + \delta_n)$ that yield a model of lower L_{exp}^{c}.

At every iteration t, the update of each λ_i in this algorithm is to take

$$\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \delta_i^{(t)}$$

where each $\delta_i^{(t)}$, $i \in \{1, ..., n\}$ satisfies

$$\delta_i^{(t)} = \frac{1}{2} \log \frac{\displaystyle\sum_{s \in S^1} s_i e^{-h(s, \Lambda^{(t)})}}{\displaystyle\sum_{s \in S^{-1}} s_i e^{h(s, \Lambda^{(t)})}}$$

We derive this update rule in Appendix A. After convergence, sentences of a new document are ranked with respect to the output of the classifier, and those with the highest scores are extracted to form the summary of the document.

An advantage of Algorithm 1 is that its complexity is linear in the number of examples, times the total number of iterations ($|\mathcal{S}| \times t$). This is interesting, since the number of sentences in the training set is generally large. In the following, we introduce our ranking framework for SDS.

2.3 Text summarisation as an ordering task

The classification framework for SDS has several drawbacks. First, the assumption that all sentences from different documents are comparable with respect to a class information is not correct. Indeed, text summaries depend more on the content of their respective documents than on a global class information. Furthermore, due to a high number of *irrelevant* sentences, a classifier will typically achieve a low misclassification rate if, independently of where relevant sentences are ranked, it always assigns the class

Algorithm 1: Classification Based Trainable Extractive Summariser

```
Input    : $\mathcal{S} = \mathcal{S}^{-1} \cup \mathcal{S}^{1}$
Initialise:
  – Normalise each sentence vector $s \in \mathcal{S}$ such that $\sum_i s_i = 1$
  – Set the value of feature weights $\Lambda^0 = (\lambda_1^0, ..., \lambda_n^0)$ with some arbitrary values

$0 \leftarrow t$
repeat
    for $i \leftarrow 1$ to $n$ do
        $\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \frac{1}{2} \log \frac{\sum_{s \in S^1} s_i e^{-h(s,\Lambda^{(t)})}}{\sum_{s \in S^{-1}} s_i e^{h(s,\Lambda^{(t)})}}$
    end
    $t \leftarrow t + 1$
until Convergence of $L^c_{exp}(\mathcal{S}; \Lambda)$ ;
Output  : $\Lambda^F$
Create a summary for each new document $d$ by taking the $n$ first sentences in $d$ with
regard to the output of the linear combination of sentence features with $\Lambda^F$
```

irrelevant to every sentence in the collection. Therefore, it is important to compare the relevance of each sentence with respect to each other within every document in the training set, in other words, to learn a ranking function that assigns higher scores to relevant sentences of a document than to irrelevant ones.

A Framework for learning a ranking function for SDS The problem of learning a trainable summariser based on ranking can be formalised as follows. For each document d in $\mathcal{D}$ we denote by S_d^1 and S_d^{-1} respectively the sets of relevant and irrelevant sentences appearing in d with respect to its summary. The ranking function can be represented by a function h that reflects the partial ordering of relevant sentences over irrelevant ones for each document in the training set. For a given document d, if we consider two sentences s and s' such that s is preferred over s' ($s \in S_d^1$ and $s' \in S_d^{-1}$) then h ranks s higher than s'

$$\forall d \in \mathcal{D}, \;\; (s, s') \in S_d^1 \times S_d^{-1} \Leftrightarrow h(s) > h(s')$$

Finally, in order to learn the ranking function we need a relevance judgment describing which sentence is preferred to which one. This information is given by binary judgments provided for documents in the training set. For these documents, sentences belonging (or not) to the summary are labeled as $+1$ (or -1).

Following [11], we can define the goal of learning a ranking function h as the minimisation of the ranking loss $L_{\mathcal{R}}$ defined as the average number of relevant sentences scored below irrelevant ones in every document d in $\mathcal{D}$

$$L_{\mathcal{R}}(h,\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_1^d||\mathcal{S}_{-1}^d|}\sum_{s\in\mathcal{S}_1^d}\sum_{s'\in\mathcal{S}_{-1}^d}[[h(s)\leq h(s')]] \tag{4}$$

Note that this formulation is similar to the misclassification error rate. The main difference, is that instead of classifying sentences as relevant/irrelevant for the summary, a ranking algorithm classifies pairs of sentences. More specifically, it considers the pair of sentences (s,s') from the same document, such that one of the two sentences is relevant. Learning a scoring function h, which gives higher score to the relevant sentence than to the irrelevant one is then equivalent to learning a classifier which correctly classifies the pair.

The Ranking Logistic Algorithm Here we are interested in the design of an algorithm which allows (a) to find efficiently a function h in the family of linear ranking functions minimising equation (4), and (b) that this function generalises well on a given test set. In this paper we address the first problem, and provide empirical evidence for the performance of our ranking algorithm on different test sets.

There exist several ranking algorithms in the ML literature, based on the perceptron [27] or *AdaBoost* - called *RankBoost* [11]. For the SDS task, as the total number of sentences in the collection may be high, we need a simple and efficient ranking algorithm. Perceptron-based ranking algorithms would lead to quadratic complexity in the number of examples, whereas the *RankBoost* algorithm in its standard setting does not search a linear combination of the input features. In this paper, we consider the class of linear ranking functions

$$\forall d\in\mathcal{D},\ \ s\in d\Rightarrow h(s,B)=\sum_{i=1}^{n}\beta_i s_i \tag{5}$$

where $B=(\beta_1,...,\beta_n)$ are the vector weights of the ranking function that we aim to learn. Similar to the explanation given in section 2.2, a logistic model is adapted to ranking[3]:

$$p(relevant|\ (s,s')) = \frac{1}{1+e^{-2\sum_{i=1}^{n}\beta_i(s_i-s'_i)}} \tag{6}$$

is well suited for learning the parameters of the combination B by minimising an exponential upper bound on the ranking loss $L_{\mathcal{R}}$, (equation 4):

$$L^r_{exp}(\mathcal{D};B) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|}\sum_{(s,s')\in\mathcal{S}_d^1\times\mathcal{S}_d^{-1}} e^{\sum_{i=1}^{n}\beta_i(s'_i-s_i)} \tag{7}$$

The interesting property of this exponential loss for ranking functions is that it can be computed in time linear to the number of examples, simply by rewriting equation (7) as follows:

$$L^r_{exp}(\mathcal{D};B) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|}\Big(\sum_{s'\in\mathcal{S}_d^{-1}} e^{\sum_{i=1}^{n}\beta_i s'_i}\Big)\Big(\sum_{s\in\mathcal{S}_d^1} e^{-\sum_{i=1}^{n}\beta_i s_i}\Big) \tag{8}$$

[3] The choice of linear ranking functions, in our case, makes it convenient to represent a pair of sentences (s,s') by the difference of their representative vectors, $(s_1-s'_1,...,s_n-s'_n)$ as $h(s)-h(s')$ becomes $\sum_{i=1}^{n}\beta_i(s_i-s'_i)$.

For the ranking case, this property makes it convenient to optimise the exponential loss rather than the corresponding binomial log-likelihood

$$\mathcal{L}_b^r(\mathcal{D};B) = -\frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|}\sum_{(s,s')\in\mathcal{S}_d^1\times\mathcal{S}_d^{-1}}\log(1+e^{-2\sum_{i=1}^n\beta_i(s_i-s'_i)}) \qquad (9)$$

Indeed, the computation of the maximum likelihood of equation (9) 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 to the number of examples through the use of the exponential loss.

For the optimisation of equation (8) we have employed the same iterative scaling procedure as in the classification case. We call our algorithm *LinearRank*, its pseudocode is shown in Algorithm 2 and its update rule ($B^{t+1} \leftarrow B^t + \Sigma^t$) is derived in Appendix B.

Algorithm 2: Ranking Based Trainable Extractive Summariser - *LinearRank*

```
Input    : $\bigcup_{d\in\mathcal{D}} \mathcal{S}_d^{-1}\times\mathcal{S}_d^1$
Initialise:
  – Normalise each sentence vector $s$ such that $\sum_i s_i = 1$, i.e. $\forall i, s_i \in [0,1]$
  – Set the value of feature weights $B^0 = (\beta_1^0, ..., \beta_n^0)$ with some arbitrary values

$0 \leftarrow t$
repeat
    for $i \leftarrow 1$ to $n$ do
        $\beta_i^{(t+1)} \leftarrow$
        $\beta_i^{(t)} + \frac{1}{2}\log\frac{\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|}\sum_{s'\in\mathcal{S}_d^{-1}}e^{h(s',B^{(t)})}\sum_{s\in\mathcal{S}_d^1}e^{-h(s,B^{(t)})}(1-s'_i+s_i)}{\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^1|}\sum_{s'\in\mathcal{S}_d^{-1}}e^{h(s',B^{(t)})}\sum_{s\in\mathcal{S}_d^1}e^{-h(s,B^{(t)})}(1+s'_i-s_i)}$
    end
    $t \leftarrow t+1$
until Convergence of $L_{exp}^r(\mathcal{D};B)$ ;
Output  : $B^F$
Create a summary for each new document $d$ by taking the $n$ first sentences in $d$ with
regard to the linear combination of sentence features with $B^F$
```

The most similar work to ours is that of Freund et al. [11] who proposed the *RankBoost* algorithm. In both cases the parameters of the combination are learnt by minimising a convex function. However, the main difference is that we propose here to learn a linear combination of the features by directly optimising equation (8), while *RankBoost* learns iteratively a nonlinear combination of the features by adaptively resampling the training data.

3 Summarising XML documents

In the following, we introduce the sentence features that we use as the input of the trainable summarisers defined in the previous section. Here, we take the logical structure of documents into account when producing summaries, as well as the content, and we learn an effective combination of features for summarisation. Although for evaluation purposes we use the INEX and SUMMAC collections, which contain scientific articles, our approach could apply to any documents formatted in XML where the logical structure is available. The summarisation of scientific texts through sentence extraction has been extensively studied in the past [33]. In our approach, we do not explicitly take advantage of the idiosyncratic nature of scientific articles, but we rather propose a generic approach that is, in essence, genre-independent. In the next section, we present the specific details of our approach.

3.1 Document features for summarisation

In this section we outline the features of XML documents that we employed in our summarisation model.

Structural features Past work on SDS (e.g. [9, 17]) has implicitly tried to take the structure of certain document types into account when extracting sentences. In [17], for example, the leading and trailing paragraphs in a document are considered important, and the position of sentences within these paragraphs is also recorded, and used, as a feature for summarisation. In our work, we move into an explicit use of structural features by taking into account the logical structure of XML documents. Our aim here is to investigate more precisely from which component of a document the summary is more likely to be generated.

The structural features we use in our approach are:

1. The depth of the element in which the sentence is contained (e.g. section, subsection, subsubsection, etc.).
2. The sibling number of the element in which the sentence is contained (e.g. 1st, middle, last).
3. The number of sibling elements of the element in which the sentence is contained.
4. The position in the element of the paragraph in which the sentence is contained (e.g. first, or not).

These features are generic, and can be applied to an entire document, or to components at any level of the XML tree that can be meaningfully summarised (i.e. components not too small to be summarised). These are just some of the features that can be used for modeling structural information; many of them have been considered for example in XML retrieval approaches (see [13]).

Content features Terms contained in the title of a document have long been recognised as effective features for automatic summarisation [9]. Our basic content-only query (COQ) comprises terms in the title of the document (*Title* query), as well as the title keywords augmented by the most frequent terms in the document (up to 10 such terms)

(*Title-MFT* query). The rationale of these approaches is that these terms should appear in sentences that are worthwhile including in summaries. The importance of title terms for SDS can also be extended to components of finer granularity (e.g. sections, subsections, etc.), by using the title of the document to find relevant sentences within any component, or, where appropriate, by using meaningful titles of components.

Since the *Title* query may be very short, sentences similar to the title which do not contain title keyword terms will have a similarity measure null with the *Title* query. To overcome this problem we have employed query-expansion techniques such as Local Context Analysis (LCA) [37] or thesaurus expansion methods (i.e. WordNet [10]), as well as a learning-based expansion technique. These three expansion techniques are described next.

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

- *Title-LCA* query, includes keywords in the title of a document and the words that occur most frequently in sentences that are most similar to the *Title* query according to the cosine measure.
- *Title-WN*, includes expanded title keywords and all their first order synonyms using WordNet.

We used the cosine measure in order to compute a preliminary score between any sentence of a document and these four queries (Title, Title-MFT, Title-LCA, Title-WN). The scoring measure doubles the cosine scoring of sentences containing acronyms (e.g. HMM (Hidden Markov Models), NLP (Natural Language Processing)), or cue-terms, e.g. ”in this paper”, ”in conclusion”, etc. The use of acronyms and cue phrases in summarisation has been emphasised in the past by [9, 17].

Learning-based expansion technique We also included two queries by forming word clusters in the document collection. This is another source of information about the relevance of sentences to summaries. It is a more contextual approach compared to the title-based queries, as it seeks to take advantage of the co-occurrence of terms within sentences all over the corpus, as opposed to the local information provided by the title-based queries.

We form different term-clusters based on the co-occurrence of words in the documents of the collection. For discovering these term-clusters, each word w in the vocabulary V is first characterised as a vector $\mathbf{w} =< n(w,d) >_{d\in\mathcal{D}}$ representing the number of occurrences of w in each document $d \in \mathcal{D}$ [4]. Under this representation, word clustering is performed using the Naive-Bayes clustering algorithm maximising the Classification Maximum Likelihood criterion [3, 29]. We have arbitrary fixed the number of clusters to $\frac{|V|}{100}$.

From these clusters, we first expand the title query by adding words which are in the same word-clusters as the title keywords. We denote this novel query by *Extended concepts with word clusters* query. Second, we represent each sentence in a document, as well as the document title, in the space of word-clusters as vectors containing the number of occurrences of words in each word-cluster in that sentence, or document title. We refer to this vector representation of document titles as *Projected concepts on word clusters* queries. The first approach (*Extended concepts with word clusters*) is a query expansion technique similar to those described above using wordnet or LCA. The second approach is a projection technique, closely related to Latent Semantic Analysis [8].

Table 1 shows some word-clusters found for the SUMMAC data collection; it can be seen from this example that each cluster can be associated to a general concept.

Word-Clusters
Cluster i: `transduction language grammar set word information model number words rules rule lexical`
Cluster j: `tag processing speech recognition morphological korean morpheme`

Table 1. An example of term clusters found for the SUMMAC data collection.

3.2 Related Work

There have been few researchers that have investigated the summarisation of information available in XML format. In [1], the work focuses on retaining the structure of the source document in the summary. A textual summary of a document is created by using lexical chains. The textual summary is then combined with the overall structure of the document with the aim of preserving the structure of the original document and of superimposing the summary on that structure. In [26], the idea of generating semantic thumbnails (essentially summaries) of documents in XML format is suggested. The authors propose to utilise the ontologies embedded in XML and RDF documents in order to develop the semantic thumbnails. Litkowski [20] has used some discourse analysis of XML documents for summarisation. In some other work [6], the tree representation of XML documents is used to generate tree structural summaries; these are summaries that focus on the structural properties of trees and do not correspond to summaries in the conventional sense of the term as used in IR research. Operations such as nesting and repetition reduction in the XML trees are used.

In the above approaches, features pertaining to the logical structure of XML documents are not taken into account when producing summaries. Structural clues are used by work on summarisation of other document types, e.g. e-mails [18], or technical documents [36]. In these summarisation approaches, known features of the structure of documents are exploited in order to produce summaries (e.g. the presence of a FAQ, or a question/answer section in technical documents).

4 Experiments

In our experiments we used 2 data sets - the INEX [13] and SUMMAC [28] test collections. For each dataset, we carried out evaluation experiments for testing (a) the query expansion effect, (b) the learning effect and the best learning scheme for SDS between classification and ranking, and (c) the effect of structure features. For point (b), we tested the performance of a linear scoring function learnt with a ranking and a classification criterion. The combination weights of the scoring function are learnt via the logistic model optimising the ranking criterion (8) by the *LinearRank* algorithm (Algorithm 2) and the classification criterion (3) using Algorithm 1. Furthermore, in order to evaluate the effectiveness of learning a linear combination of sentence features for SDS under the ranking framework, we compared the performance of the *LinearRank* algorithm and the RankBoost algorithm [11] which learn a non-linear combination of features. To measure the effect of structure features, we have learnt the best learning algorithm using COQ features alone, and using COQ features together with the structure features.

4.1 Datasets

We used version 1.4 of the INEX document collection. This version consists of 12,107 articles of the IEEE Computer Society's publications, from 1995 to 2002, totaling 494 megabytes. It contains over 8.2 million element nodes of varying granularity, where the average depth of a node is 6.9 (taking an article as the root of the tree). The overall structure of a typical article consists of a front matter (containing e.g. title, author, publication information and abstract), a body (consisting of e.g. sections, sub-sections, sub-sub-sections, paragraphs, tables, figures, lists, citations) and a back matter (including bibliography and author information).

The SUMMAC corpus consists of 183 articles. Documents in this collection are scientific papers which appeared in ACL (Association for Computational Linguistics) sponsored conferences. The collection has been marked up in XML by converting automatically the latex version of the papers to XML. In this dataset the markup includes tags covering information such as title, authors or inventors, etc., as well as basic structure such as abstract, body, sections, lists, etc.

We have removed documents from the INEX dataset that do not possess title keywords or an abstract. From the SUMMAC dataset, we removed documents whose title contained no-informative words, such as a list of proper names. From each dataset, we also removed documents having extractive summaries (as found by Marcu's algorithm, see Section 4.2) composed of one sentence only, arguing that a sentence is not sufficient to summarise a scientific article. In our experiments, we used in total 161 documents from SUMMAC and $4,446$ documents from INEX collections.

We extracted the logical structure of XML documents using freely available structure parsers. Documents are tokenised by removing words in a stop list, and sentence boundaries within each document are found using the morpho-syntactic tree tagger program [35]. In Table 2, we show some statistics about the two document collections used, about the abstracts provided with the two collections, and about the extracts that were created using Marcu's algorithm, as well as the training/test splits for each dataset (in all experiments the size of the training and test sets are kept fixed). Both datasets have roughly the same characteristics of sentence distribution in the articles and summaries. The summary length, in number of sentences, is approximately 9 and 6 in average for the Summac and INEX collections respectively.

Data set comparison		
Source	SUMMAC	INEX
Number of docs	161(183)	4446(12107)
Training/Test splits	80/81	1000/3446
Total # of sentences in the collection	24725	817175
Average # of sentence per doc.	153.57	183.8
Maximum # of sentence per doc.	799	864
Minimum # of sentence per doc.	6	12
Average # of words per sentence	11.39	14.12
Size of the vocabulary	15119	153422
Average extract size (in # of sentence)	9.71	6.07
Maximum # of sentence per extract	36	37
Minimum # of sentence per extract	2	3
Average abstract size (in # of sentence)	9.2	5.92
Maximum # of sentence per abstract	24	25
Minimum # of sentence per abstract	4	5

Table 2. Data set properties.

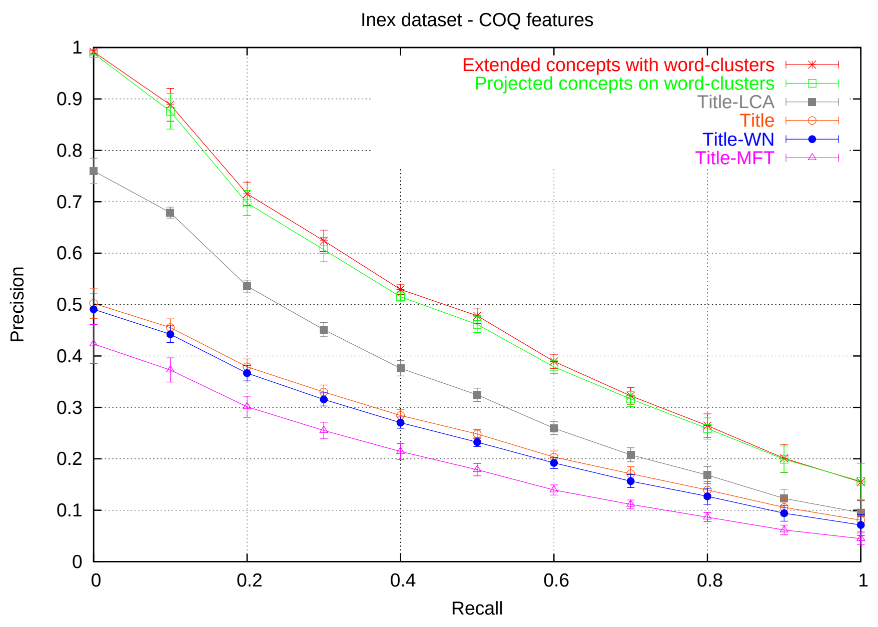
4.2 Experimental Setup

We assume that for each document, summaries will only include sentences between the introduction and the conclusion of the document. A compression ratio must be specified for extractive summaries. For both datasets we followed the SUMMAC evaluation by using a 10% compression ratio [28].

To obtain sentence-based extract summaries for all articles in both datasets, for training and evaluation purposes, we need *gold* summaries. The human extraction of such reference summaries, in the case of large datasets, is not possible. To overcome this restriction we use in our experiments the author-supplied abstracts that are available with the original articles, and apply an algorithm proposed by Marcu [22] in order to generate extracts from the abstracts. This algorithm has shown a high degree of correlation to sentence extracts produced by humans. We therefore evaluate the effectiveness of our learning algorithm on the basis of how well it matches the automatic extracts.

The learning algorithms take as input the set of features defined in section 3.1. Each sentence in the training set is represented as a feature vector, and the algorithms are learnt based on this input representation and the extracted summaries found by Marcu's algorithm [22], which were used as desired outputs.

For all the algorithms, on each dataset, we have generated precision and recall curves to measure the query expansion and learning effects. Precision and recall are



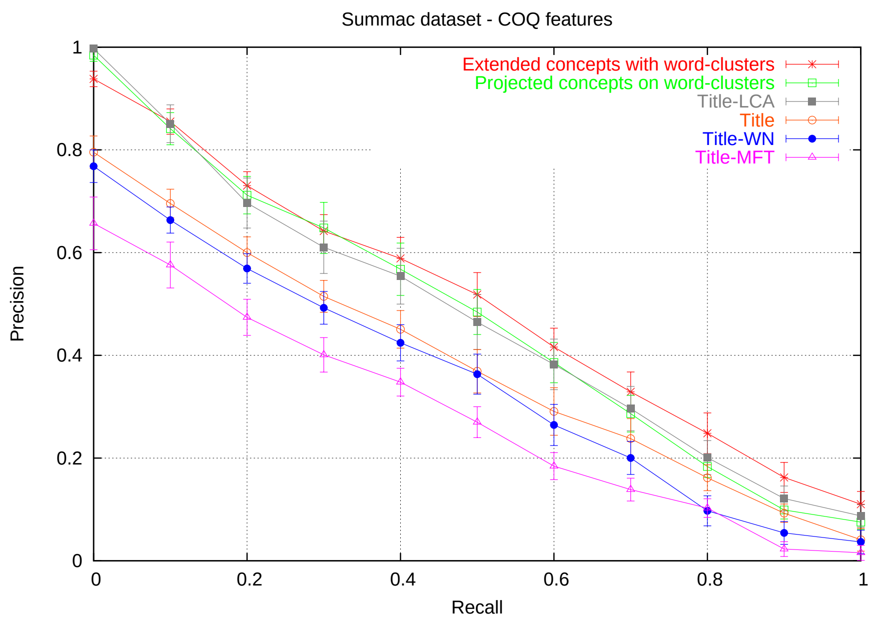


Fig. 2. Precision-Recall curves at 10% compression ratio for the COQ features on INEX (top) and SUMMAC (bottom) datasets. Each point represents the mean performance for 10 cross-validation folds. The bars show standard deviations for the estimated performance.

computed as follows:

$$\text{Precision} = \frac{\text{\# of sentences in the extract and also in the gold standard}}{\text{total \# of sentences in the extract}}$$
$$\text{Recall} = \frac{\text{\# of sentences in the extract and also in the gold standard}}{\text{total \# of sentences in the gold standard}}$$

Precision and recall values are averaged over 10 random splits of the training/test sets. We have also measured the break-even point at 10% compression ratio for the 3 learning algorithms and the best COQ feature (Table 3).

5 Analysis of Results

We examine the results from three viewpoints: in Section 5.1 we present the effectiveness of each of the content only queries (COQ) alone, as well as the query expansion effect, in Section 5.2 we examine the performance of the three learning algorithms, and in Section 5.3 we look into the effectiveness of our summarisation approach for XML documents.

5.1 Query expansion effects

In Figure 2, we present the precision and recall graphs showing the effectiveness of content-only features for SDS without the learning effect (i.e. by using each content feature individually to rank the sentences). The order of effectiveness of the features seems to be consistent across the two datasets: extended concepts with word clusters are the most effective, followed by projected concepts on word clusters and title with local context analysis. Title with the most frequent terms in the document is the least effective feature in both cases.

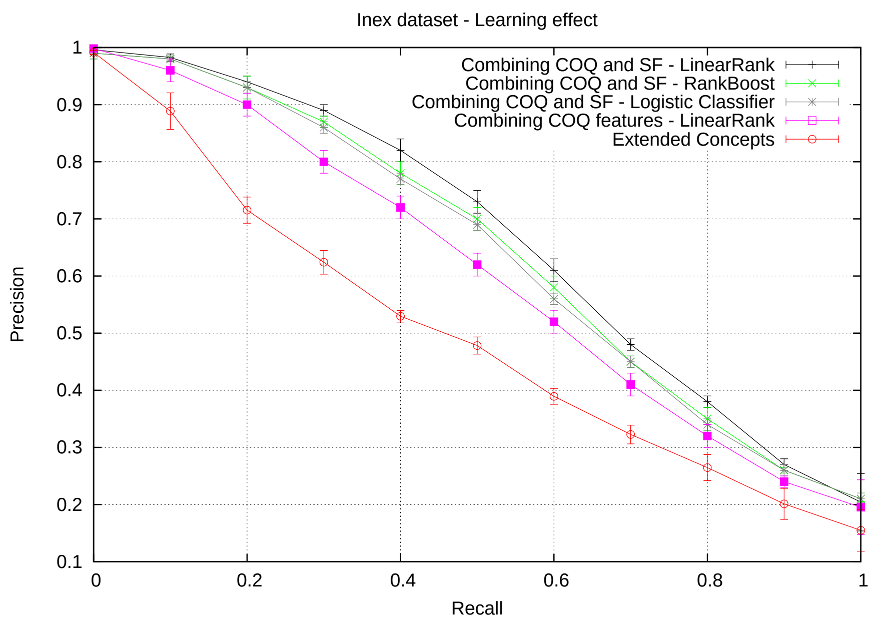
The high effectiveness obtained with word clusters (extended and projected concepts with word clusters) demonstrates that the contextual approach investigated here is effective and should be further exploited for SDS.

5.2 Learning algorithms

In Figure 3, we present the precision and recall graphs obtained through the combination of content and structure features for the two datasets when using the three learning algorithms. For comparison, we display the Precision-Recall curves obtained for the best CO feature (Extended concepts) with those obtained from the learning algorithms.

A first result is that the combination of features by learning outperforms each feature alone. The results also show that the two ordering algorithms are more effective in both datasets than the logistic classifier. This finding corroborates with the justification given in Section 2.3.

When comparing the two ordering algorithms, we see that Algorithm 2 (Linear-Rank) slightly outperforms the RankBoost algorithm for low recall values. Since both



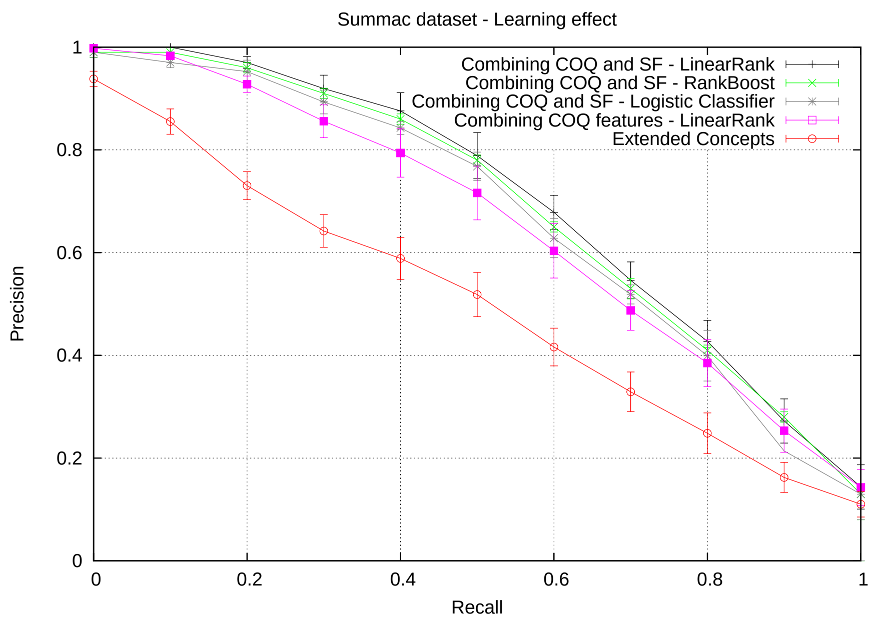


Fig. 3. Precision-Recall curves at 10% compression ratio for the learning effects on INEX (top) and SUMMAC (bottom) datasets.

ordering algorithms optimise the same criteria (equation 8), the difference in performance can be explained by the class of functions that each algorithm learns. The RankBoost algorithm outputs a nonlinear combination of the features, while with the *LinearRank* algorithm we obtain a linear combination of these features. As the space of features is small, the non-linear RankBoost model has low bias and high variance and hence attempts to overfit the data. We have noticed this effect in both test collections by comparing Precision and Recall curves for RankBoost on the test and the training sets.

Our experimental results suggest that a ranking criterion is better suited to the SDS task than a classification criterion. Moreover, a simple logistic model performs better than a non-linear algorithm and, depending on the implementation, can be significantly faster to train than RankBoost. This leads to the conclusion that such a linear model, i.e. optimising equation (8), can be a good choice for learning a summariser, in particular when considering structural features.

5.3 Summarisation effectiveness

By looking at the data in Figure 3 from the point of view of comparing the effectiveness of the summariser with different features, one can note that the combination of content and structure features yields greater effectiveness than the use of content features alone. This result seems to hold equally for both document sets for most recall points. In terms of break-even points, (Table 3), the increase in effectiveness is approximately 3% for the RankBoost and *LinearRank* algorithms in both data sets[4]. This provides evidence that the use of structural features improves the effectiveness of the task of SDS.

It is to be noted that as the structural features we considered here are discrete, the ordering of sentences with respect to different structural components was, hence, not possible. Training the learning models using only these features did not provide significant results either (we chose not to display these results as they were not informative). The fact that structural features increase the performance of the learning models when they are added to CO features, is in our opinion due to that structural features provide non-redundant information compared to CO features.

Break-even points (%)							
Data sets	Best COQ	Classifier		RankBoost		LinearRank	
		COQ features	COQ+SF	COQ features	COQ+SF	COQ features	COQ+SF
SUMMAC	50.9	56.4	61.05	57.8	62.1	59.1	63.2
INEX	48.5	55.1	58	56.1	59.1	57.06	60.6

Table 3. Break-even points at 10% compression ratio for learning algorithms and the best COQ feature: Extended title keywords with word-clusters. Each value represents the mean performance for 10 cross-validation folds.

From the set of structure features used in our experiments (Section 3.1), the depth of the sentence's component and the paragraph's position containing summary sentences

[4] The same performance increase is also obtained from the classifier.

within the component (i.e. whether it is in the first paragraph or not of a component) got the highest weights with both ranking algorithms. Any sentence in the first paragraph of any first sections of a document, containing relevant COQ features, thus got high scores. In our experiments, these two structural features were the most effective for SDS. It is well known that, in scientific articles, sentences in the first parts of sections such as Introduction and Conclusions are useful for summarisation purposes [9, 17]. Our results agree with this, as the increased weights for the paragraph's position in a component suggests. The features corresponding to the position of elements with respect to their siblings are less effective than depth and paragraph position, but features indicating the position of an element as the first or the last sibling have a higher impact than when the element was the middle sibling. We should also note that the feature corresponding to the number of siblings of an element was the least conclusive in all of our experiments; its utility seemed to highly depend on the dataset.

For the specific case of scientific text, from the set of structure features used, a set of features which is known to be effective was weighted higher by our summarisation method. One way to view this result is that our method correctly identified features that are known to be effective for this document genre, and has therefore the potential to perform equally well in other document genre. This in turn, can be seen as an indication that the use of structure features could be applied to document collections of different genre. The availability of suitable document collections containing different document types will be necessary in order to test this assertion.

By looking at the data in Table 3 (and Figures 2 and 3), one can note that effectiveness when using the INEX collection is always lower than when using the SUMMAC collection. This difference in effectiveness can be attributed to the different characteristics of the two datasets. The INEX collection contains many more documents than SUMMAC, and is also a more heterogeneous dataset. In addition, the logical structure of INEX documents is more complex than that of the SUMMAC collection. These factors are likely to cause the small difference in effectiveness between the two collections.

6 Discussion and conclusions

The results presented in the previous section are encouraging in relation to our two main motivations: a novel learning algorithm for SDS, and the inclusion of structure, in addition to content, features for the summarisation of XML documents. In terms of the algorithms, it was shown that using the same logistic model, but choosing a ranking criterion instead of a classification one, leads to a notable performance increase. Moreover, compared to `RankBoost`, the `LinearRank` algorithm performs better and it also has the potential to be implemented in a simpler manner. This property may make this latter algorithm an effective and efficient choice for the task of SDS.

In terms of the summarisation of XML documents by using content and structure features, the results demonstrated that for both datasets, the inclusion of structural features improve the effectiveness of learning algorithms for SDS. The improvements are not dramatic, but they are consistent across both datasets and across most recall points. This consistency suggests that the inclusion of features from the logical structure of XML documents is effective.

The ultimate aim of our approach for the summarisation of XML documents is to produce summaries for components at any level of granularity (e.g. section, subsection, etc.). The content and structure features that we presented in Section 3.1 can be applied to any level of granularity. For example, the depth of an element, the sibling number of an element in which a sentence is contained, the number of sibling elements in which the sentence is contained, and the position in the element of the paragraph in which the sentence is contained (i.e. the structure features in section 3.1) can be applied to entire documents, sections, subsections, etc. Essentially, they can be applied to any XML element that can be meaningfully summarised, i.e. that is informative and long enough to make its summarisation meaningful [31]. In particular, the most effective content (expanded concepts with word clusters and projected concepts on word clusters), and structure features (depth of element and position of paragraph in the element), can be applied to various granularity levels within an XML tree. The effectiveness of such an approach however, cannot be tested until datasets with human produced summaries, or summary extracts, at component level become available. We should also note that we focus on generic (rather than query-biased) summaries for evaluation purposes, but the proposed model can be applied to both types of summarisation.

In Section 5.3 we mentioned that the results provide us with some indication that the use of structural features can also be effective for summarising XML documents from datasets containing documents other than scientific articles. One possible direction for future research would therefore be to examine this issue in more detail, and to identify appropriate datasets of non-scientific XML data for summarisation. The list of structural features that we use in this study is short, so a larger variety of features could be investigated. When moving into document collections of different types, it will be worthwhile to investigate whether useful structural features can be derived automatically, e.g. by looking at a collection's DTD.

Some further interesting issues that arise when considering the summarisation at any structural level, relate to the choice of the appropriate components to be summarised. For example, it may be unrealistic to provide summaries of very small size components, or of components that are not informative enough. One of the main research issues in XML retrieval is to define and understand what a meaningful retrieval unit is [13]. One direction to follow, would be to conduct a user study in which to observe what kinds of XML elements searchers would prefer to see in a summarised version after the initial retrieval. Some initial investigation can be found in [30, 31], where results indicate a positive correlation between element probability of relevance, length and user preference to see summary information. Further research in this direction is currently underway.

By looking at the results of this study as a whole, we can say that the work presented here achieved its main aim, to effectively summarise XML documents by combining content and structure features through using novel machine learning approaches. Both datasets that we used contain scientific articles, that have some inherent characteristics which may simplify the task of SDS. This work has however a greater impact, as we believe that it can be applied to datasets containing documents of other types. The availability of XML data will continue to increase as, for example, XML is becoming the W3C standard for representing documents (e.g. in digital libraries where content can be of any type). The availability of intelligent summarisation approaches for XML data with therefore become increasingly important, and we believe that this work has provided a step towards this direction.

References

1. H. Alam, A. Kumar, M. Nakamura, F. Rahman, Y. Tarnikova, and C. Wilcox. Structured and unstructured document summarization: Design of a commercial summarizer using lexical chains. In *Proceedings of the* 7^{th} *International Conference on Document Analysis and Recognition*, pages 1147–1152. IEEE, 2003.
2. M.-R. Amini and P. Gallinari. The use of unlabeled data to improve supervised learning for text summarization. In *Proceedings of the* 25^{th} *ACM SIGIR Conference*, pages 105–112, 2002.
3. M.-R. Amini, N. Usunier, and P. Gallinari. Automatic text summarization based on word clusters and ranking algorithms. In *Proceedings of the* 27^{th} *European Conference on Information Retrieval*, pages 142–156, 2005.
4. M. Caillet, J.-F. Pessiot, M.-R. Amini, and P. Gallinari. Unsupervised learning with term clustering for thematic segmentation of texts. In *Proceedings of the 7th Recherche d'Information Assiste par Ordinateur, Avignon, France*, pages 648–656. CID, 2004.
5. W. T. Chuang and J. Yang. Extracting sentence segments for text summarization: a machine learning approach. In *Proceedings of the* 23^{rd} *ACM SIGIR Conference*, pages 152–159, 2000.
6. T. Dalamagas, T. Cheng, K.-J. Winkel, and T. K. Sellis. Clustering xml documents using structural summaries. In *Proceedings of the EDBT Workshop on Clustering Information over the Web*, pages 547–556, 2004.
7. J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.
8. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
9. H. Edmundson. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285, 1969.
10. C. D. Fellbaum. *WordNet, an Electronic Lexical Database*. MIT Press, Cambridge MA, 1998.
11. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.
12. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. volume 38, pages 337–374, 2000.
13. N. Fuhr, S. Malik, and M. Lalmas. Overview of the initiative for the evaluation of xml retrieval (inex) 2003. In *Proceedings of the Second INEX Workshop*, 2004.
14. U. Hahn and I. Mani. The challenges of automatic summarization. In *IEEE Computer Society*, volume 33, pages 29–36, 2000.
15. T. Hirao, H. Isozaki, E. Maeda, and Y. Matsumoto. Extracting important sentences with support vector machines. In *The* 19^{th} *International Conference on Computational Linguistics*, 2002.
16. J. Hutchins. Summarization: Some problems and methods. In K. Jones, editor, *Meaning: The Frontier of Informatics*, volume 9, pages 151–173. Aslib, 1987.
17. J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the* 18^{th} *ACM SIGIR Conference*, pages 68–73, 1995.

18. D. Lam, S. Rohall, C. Schmandt, and M. Stern. Exploiting e-mail structure to improve summarization. *Technical Report 02-02, IBM Watson Research Centre*, 2002.
19. A. M. Lam-Adesina and G. J. F. Jones. Applying summarization techniques for term selection in relevance feedback. In *Proceedings of the 24^{th} ACM SIGIR Conference*, pages 1–9, 2001.
20. K. Litkowski. Text summarization using xml-tagged documents. In *Proceedings of the Document Understanding Conference (DUC'2003)*, pages 58–65, 2003.
21. H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal*, 2:159–165, 1958.
22. D. Marcu. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the 22nd ACM SIGIR Conference*, pages 137–144, 1999.
23. C. Paice. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management*, 26(1):171–186, 1990.
24. T. Sakai and K. S. Jones. Generic summaries for indexing in information retrieval. In *Proceedings of the 24^{th} ACM SIGIR Conference*, pages 190–198, 2001.
25. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
26. A. Sengupta, M. Dalkilic, and J. Costello. Semantic thumbnails: a novel method for summarizing document collections. In *Proceedings of the 22^{nd} annual international conference on Design Of Communication*, pages 45–51. ACM Press, 2004.
27. L. Shen and A. K. Joshi. Ranking and reranking with perceptron. *Machine Learning, Special Issue on Learning in Speech and Language Technologies*, 60(1-3):73–96, 2005.
28. SUMMAC. http://www.itl.nist.gov/iaui/894.02/related_projects/ tipster_summac/cmp_lg.html.
29. M. Symons. Clustering criteria and multivariate normal mixture. *Biometrics*, 37:35–43, 1981.
30. Z. Szlavik, A. Tombros, and M. Lalmas. Investigating the use of summarisation for interactive xml retrieval. In *Proceedings of the 21^{st} ACM Symposium on Applied Computing, Information Access and Retrieval Track*, pages 1068–1072, 2006.
31. Z. Szlavik, A. Tombros, and M. Lalmas. The use of summaries in xml retrieval. In *Proceedings of the 10^{th} European Conference on Research and Advanced Technology for Digital Libraries*, pages 75–86, 2006.
32. S. Teufel and M. Moens. Sentence extraction as a classification task. In *Proceedings of the Intelligent Scalable Text Summarization Workshop, ACL*, pages 58–65, 1997.
33. S. Teufel and M. Moens. Summarizing scientific articles – experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445, 2002.
34. A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21^{st} ACM SIGIR Conference*, pages 2–10, 1998.
35. TreeTagger. http://www.ims.uni-stuttgart.de/projekte/corplex/treetagger/ decisiontreetagger.html.
36. C. Wolf, S. Alpert, J. Vergo, L. Kozakov, and Y. Doganata. Summarizing technical support documents for search: expert and user studies. *IBM Systems Journal*, 43(3):564–586, 2004.
37. J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19^{th} ACM SIGIR Conference*, pages 4–11, 1996.

Appendix

In this section we derive the update rules for iterative scaling given in Algorithms 1 and 2. For further details about the iterative scaling approach, such as proof of convergence, please refer to [7].

A. Minimising the exponential loss for classification (Algorithm 1)

We aim to find a procedure $\Lambda \leftarrow \Lambda + \Delta$ which takes one set of parameters as input and produces a new set as output that decreases the exponential loss L^c_{exp} for the classification case. We apply the transformation until we reach a stationary point for Λ. The change in the exponential loss (3) from the set Λ to the set $\Lambda + \Delta$ is

$$L^c_{exp}(\Lambda + \Delta) - L^c_{exp}(\Lambda) = \frac{1}{|\mathcal{S}|}\left[\sum_{y\in\{-1,1\}}\sum_{\mathbf{s}\in\mathcal{S}^y}\left\{e^{-yh(s,\Lambda)}\left(e^{\sum_{i=1}^n -y\delta_i s_i} - 1\right)\right\}\right]$$

We suppose here that sentence features are normalised and are all positive values: $\forall i, s_i \geqslant 0$ and $\sum_i s_i = 1$. Hence, by Jensen's inequality applied to e^x we have

$$L^c_{exp}(\Lambda + \Delta) - L^c_{exp}(\Lambda) \leqslant \frac{1}{|\mathcal{S}|}\left[\sum_{y\in\{-1,1\}}\sum_{\mathbf{s}\in\mathcal{S}^y}\left\{e^{-yh(s,\Lambda)}\left(\sum_{i=1}^n s_i e^{-y\delta_i} - 1\right)\right\}\right]$$

Let us denote the right hand side of the inequality by $\mathcal{A}$

$$\mathcal{A}(\Lambda,\Delta) = \frac{1}{|\mathcal{S}|}\sum_{y\in\{-1,1\}}\sum_{\mathbf{s}\in\mathcal{S}^y}\left\{e^{-yh(s,\Lambda)}\left(\sum_{i=1}^n s_i e^{-y\delta_i} - 1\right)\right\}$$

Since $L^c_{exp}(\Lambda+\Delta)-L^c_{exp}(\Lambda) \leqslant \mathcal{A}(\Lambda,\Delta)$, then if we can find a Δ for which $\mathcal{A}(\Lambda,\Delta) < 0$, then the new set of parameters $\Lambda + \Delta$ is an improvement (in terms of the exponential loss) over the initial parameters Λ. A greedy strategy for optimising the parameters of the logistic classifier is to find the Δ which minimises $\mathcal{A}(\Lambda,\Delta)$, set $\Lambda \leftarrow \Lambda + \Delta$, and repeat. Here, we proceed by finding the stationary point of the auxiliary function $\mathcal{A}$ with respect to Δ:

$$\forall i, \frac{\partial\mathcal{A}(\Lambda,\Delta)}{\partial\delta_i} = \sum_{y\in\{-1,1\}}\sum_{\mathbf{s}\in\mathcal{S}^y} e^{-yh(s,\Lambda)}(-ys_i)e^{-y\delta_i} = 0$$

which is equivalent to

$$\begin{aligned} &\forall i, \sum_{\mathbf{s}\in\mathcal{S}^{-1}} e^{h(s,\Lambda)} s_i e^{\delta_i} - \sum_{\mathbf{s}\in\mathcal{S}^1} e^{-h(s,\Lambda)} s_i e^{-\delta_i} = 0 \\ \Leftrightarrow\ &\forall i, e^{2\delta_i}\sum_{\mathbf{s}\in\mathcal{S}^{-1}} s_i e^{h(s,\Lambda)} = \sum_{\mathbf{s}\in\mathcal{S}^1} e^{-h(s,\Lambda)} s_i \end{aligned}$$

The update rule is then to iteratively add to the current parameter set the parameters

$$\forall i, \delta_i = \frac{1}{2}\log\frac{\displaystyle\sum_{s\in S^1} s_i e^{-h(s,\Lambda)}}{\displaystyle\sum_{s\in S^{-1}} s_i e^{h(s,\Lambda)}}$$

B. Minimising the exponential loss for ranking (Algorithm 2)

For the update rule $B \leftarrow B + \Sigma$, we assume that every component i of Σ gets updated separately. As each sentence feature takes values in $[0,1]$, for each feature component i of each pair $(s,s') \in \mathcal{S}_d^{-1} \times \mathcal{S}_d^{1}$ we have $s'_i - s_i \in [-1,1]$.

$$L^r_{exp}(B+\sigma_i) - L^r_{exp}(B) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|}\left[\sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)} \sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}\left[e^{\sigma_i(s'_i-s_i)} - 1\right]\right]$$
$$= \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|}\left[\sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)} \sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}\left[e^{(\frac{1+(s'_i-s_i)}{2})\sigma_i + (\frac{1-(s'_i-s_i)}{2})(-\sigma_i)} - 1\right]\right]$$

By Jensen's inequality applied to e^x we have

$$e^{(\frac{1+(s'_i-s_i)}{2})\sigma_i + (\frac{1-(s'_i-s_i)}{2})(-\sigma_i)} \leq \left(\frac{1+(s'_i-s_i)}{2}\right)e^{\sigma_i} + \left(\frac{1-(s'_i-s_i)}{2}\right)e^{-\sigma_i}$$

From this inequality it follows that

$$L^r_{exp}(B+\Sigma) - L^r_{exp}(B) \leq \frac{1}{|\mathcal{D}|}\mathcal{E}(B,\sigma_i)$$

where,

$$\mathcal{E}(B,\sigma_i) = \sum_{d\in\mathcal{D}} \frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|} \sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)} \sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}\left[\left(\frac{1+(s'_i-s_i)}{2}\right)e^{\sigma_i} + \left(\frac{1-(s'_i-s_i)}{2}\right)e^{-\sigma_i} - 1\right]$$

The stationary point of $\mathcal{E}$ with respect to σ_i is then

$$\sum_{d\in\mathcal{D}} \tfrac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|} \textstyle\sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)} \sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}\left[\left(\tfrac{1+(s'_i-s_i)}{2}\right)e^{\sigma_i} + \left(\tfrac{(s'_i-s_i)-1}{2}\right)e^{-\sigma_i}\right] = 0$$

$$\Rightarrow \qquad \sigma_i = \tfrac{1}{2}\log\frac{\displaystyle\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|}\sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)}\sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}(1-s'_i+s_i)}{\displaystyle\sum_{d\in\mathcal{D}}\frac{1}{|\mathcal{S}_d^{-1}||\mathcal{S}_d^{1}|}\sum_{s'\in\mathcal{S}_d^{-1}} e^{h(s',B)}\sum_{s\in\mathcal{S}_d^{1}} e^{-h(s,B)}(1+s'_i-s_i)}$$