Machine Learning Ranking for Structured Information Retrieval

Jean-Noël Vittaut and Patrick Gallinari

Laboratoire d'Informatique de Paris 6,
8, rue du Capitaine Scott, F-75015 Paris, France
{vittaut, gallinari}@poleia.lip6.fr

Abstract. We consider the Structured Information Retrieval task which consists in ranking nested textual units according to their relevance for a given query, in a collection of structured documents. We propose to improve the performance of a baseline Information Retrieval system by using a learning ranking algorithm which operates on scores computed from document elements and from their local structural context. This model is trained to optimize a Ranking Loss criterion using a training set of annotated examples composed of queries and relevance judgments on a subset of the document elements. The model can produce a ranked list of documents elements which fulfills a given information need expressed in the query. We analyze the performance of our algorithm on the INEX collection and compare it to a baseline model which is an adaptation of Okapi to Structured Information Retrieval.

1 Introduction

Structured document collections, with documents encoded into a structured representation standard such as XML, XHTML, RDF, RSS are now becoming available and the IR community has started to develop search engines specifically dedicated to this type of documents [1] [2]. Document structure offers many new possibilities such as answering queries with structural constraints (Content and Structure queries in the INEX context[1] [1]), or simply providing the user with a list of relevant units with different granularities (Content Only queries in INEX (CO)). These units may correspond to different types of document elements in a structured document. In this paper, we consider the latter (CO) problematic. One difficulty of this task is to compare and rank document elements with very different characteristics such as their length, their redundancy, their thematic homogeneity, etc. Traditional search engines have been developed for ranking similar documents and are not adapted to this ranking task. Different frameworks have been developed for scoring elements in structured documents. For example theory of evidence has been used for aggregating evidence from sub-documents elements [3] [4]. Bayesian networks [5] or language models [6] are other formal paradigms for combining evidence from sub-elements in order

[1] INEX is the "INitiative for the Evaluation of XML Retrieval of the DELOS network of excellence".

M. Lalmas et al. (Eds.): ECIR 2006, LNCS 3936, pp. 338–349, 2006.

to score a containing document element. In the Machine Learning community, ranking algorithms have recently motivated different studies and developments. In the field of textual documents, they have been successfully used to combine features or preferences relations in task such as meta search [7] [8] [9], automatic summarization [10] and recently for the combination of different sources of evidence in IR [11]. One of the difficulties of this paradigm is its complexity which is in the general case quadratic in the number of examples. Linear solutions have been proposed by some authors [9] [10]. Our approach extends the work of [10] to multiclass problems whereas most of experiments of ranking are based on two-classes problems, with a small class of relevant elements, and a large class of irrelevant elements. Under some conditions, fast rates of convergence are achieved with this class of methods [12].

We propose here to develop and use ranking methods adapted to a particular task of Structured Information Retrieval (SIR) which consists in producing a list of ordered documents elements which fulfills a content oriented query. Ranking can be particularly useful for SIR due to the intrinsic difficulty of this task, as already mentioned above, and because traditional search engines are not well adapted to this ranking task. It is hoped that ranking algorithms may help to improve the performance of existing techniques. Ranking algorithms work by combining features which characterize the data elements to be ranked. In our case, these features will depend on the document element itself and on its structural context. Ranking algorithms will learn to combine these different features in an optimal way according to a specific loss Function using a set of examples.

The paper is organized as follows, in section 2 we present the ranking model, in section 3 we show how it can be adapted to Structured Information Retrieval. In section 4 we describe some experiments with content-based queries on a semi-structured database and compare the algorithm to a baseline Okapi method adapted for SIR.

2 Framework

We present in this section a general model of ranking which can be adapted to IR or SIR. The idea of the ranking algorithms proposed in the Machine Learning community is to learn a total order on a set $\mathcal{X}$, which allows to compare any element pair in this set. Given this total order, we are able to order any subset of $\mathcal{X}$ in a ranking list. For instance in IR, $\mathcal{X}$ can be the set of documents which are relevant to a given query, and the total order is the natural order on the document scores.

As for any Machine Learning technique, one needs a training set of labeled examples in order to learn how to rank. This training set will consist in ordered pairs of examples. This will provide a partial order on the elements of $\mathcal{X}$. The ranking algorithm will use this information to learn a total order on the elements of $\mathcal{X}$ and after that will allow to rank new elements. For plain IR, the partial ordering may be provided by human assessments on different documents for a given query.

We introduce below some notations which will be used to compare the different subsets of a partially ordered set $\mathcal{X}$.

2.1 Notations

Let $\mathcal{X}$ be a set of elements with a partial order $\prec$ defined on it. This means that some of the element pairs in $\mathcal{X}$ may be compared according to the $\prec$ relation. If there is no preference between two element x and x', this is denoted by $x \perp x'$. For Structured Information retrieval $\mathcal{X}$ will be the set of couples (doxel[2], query) for all doxels and queries in the document collection. This set is partially ordered according to the existing relevance judgments for each query.

The set of all couples of $\mathcal{X} \times \mathcal{X}$ which are comparable according to $\prec$ is denoted:

$$\mathcal{A}(\mathcal{X}) = \{(x, x') \in \mathcal{X} \times \mathcal{X} / x \prec x' \text{ or } x' \prec x\}.$$

2.2 Ranking

Let f be a Function from $\mathcal{X}$ to the set of real numbers. We can associate a total order to f such that:

$$x \prec x' \Leftrightarrow f(x) < f(x') \ . \tag{1}$$

Clearly, learning the f Function is the same as learning the total order.

An element of $\mathcal{X}$ is represented by a real vector of features $x = (t_1, t_2, ..., t_d)$. In our case, the features will be local scores computed on different contextual elements of a doxel (label, parent, children, document...). In the following, f will be a linear combination of x's features:

$$f_\omega(x) = \sum_{j=1}^{d} \omega_j t_j \tag{2}$$

where $\omega = (\omega_1, \omega_2, ..., \omega_d)$ are the parameters of the combination to be learned.

Ranking Loss. f_ω is said to respect $x \prec x'$ if $f_\omega(x) < f_\omega(x')$. In this case, couple (x, x') is said to be well ordered by f_ω. The ranking loss [9] measures how much f_ω respects $\prec$.

By definition, the ranking loss measures the number of mis-ordered couples in $\mathcal{A}(\mathcal{X})$:

$$R(\mathcal{A}(\mathcal{X}), \omega) = \sum_{\substack{(x,x') \in \mathcal{A}(\mathcal{X}) \\ x \prec x'}} \chi(x, x') \tag{3}$$

where $\chi(x, x') = 1$ if $f_\omega(x) > f_\omega(x')$ and 0 otherwise.

Ranking aims at learning ω for minimizing Function 3. This approach is different from previous Machine Learning approaches to IR such as Logistic Regression [13], since we do not try to classify elements into relevant group and irrelevant groups. Here, we are only interested in producing a well ordered list. That is why the criterion we use is only based on the relative position of elements in the

[2] Doxel means *document element*, a subpart of the document.

ranking list rather than on an absolute probability of belonging to a relevant or irrelevant class.

Exponential loss. In practice, this expression is not very useful since χ is not differentiable, ranking algorithms use to optimize an approximation of this loss criterion called the exponential loss:

$$R_e(\mathcal{A}(\mathcal{X}), \omega) = \sum_{\substack{(x,x') \in \mathcal{A}(\mathcal{X}) \\ x \prec x'}} e^{f_\omega(x) - f_\omega(x')} \tag{4}$$

It is straightforward that $R(\mathcal{A}(\mathcal{X}), \omega) \leq R_e(\mathcal{A}(\mathcal{X}), \omega)$. Function 4 is differentiable and convex, and then can be minimized using standard optimization techniques. Minimizing Function 4 will allow to minimize $R(\mathcal{A}(\mathcal{X}), \omega)$.

Properties. Some properties may be inferred from Function 4 which will allow us to reduce the complexity of the learning algorithm. In the general case, the complexity is quadratic in the number of examples and this does not allow us to learn with more than a few thousands of examples, which is not sufficient for most of real tasks.

Let us introduce on the subsets of $\mathcal{X}$, $\mathcal{P}(\mathcal{X})$, a derived strict partial order $\prec_{\mathcal{P}(\mathcal{X})}$ from $\prec$. For two subsets $\mathcal{X}_i$ and $\mathcal{X}_j$ of $\mathcal{X}$:

$$\mathcal{X}_i \prec_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j \Leftrightarrow \begin{cases} \forall x_i \in \mathcal{X}_i, \forall x_j \in \mathcal{X}_j : x_i \prec x_j \text{ and} \\ \forall (x_i, x_i') \in \mathcal{X}_i \times \mathcal{X}_i : x_i \perp x_i' \text{ and} \\ \forall (x_j, x_j') \in \mathcal{X}_j \times \mathcal{X}_j : x_j \perp x_j' \end{cases}$$

We derive similarly $\perp_{\mathcal{P}(\mathcal{X})}$ from $\perp$ as :

$$\mathcal{X}_i \perp_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j \Leftrightarrow \forall x_i \in \mathcal{X}_i, \forall x_j \in \mathcal{X}_j : x_i \perp x_j.$$

Using these definitions, we can deduce elementary properties for $R_e(\mathcal{A}(\mathcal{X}), \omega)$.

Property 1. We denote $\mathcal{A}_i = \mathcal{A}(\mathcal{X}_i)$ and $\mathcal{A}_j = \mathcal{A}(\mathcal{X}_j)$. If any element of $\mathcal{X}_i$ is not comparable to any element of $\mathcal{X}_j$, the ranking loss can be expressed as the sum of two sub-ranking losses over $\mathcal{A}(\mathcal{X}_i)$ and $\mathcal{A}(\mathcal{X}_j)$:

$$\mathcal{X}_i \perp_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j \Rightarrow R_e(\mathcal{A}(\mathcal{X}_i) \cup \mathcal{A}(\mathcal{X}_j), \omega) = R_e(\mathcal{A}(\mathcal{X}_i), \omega) + R_e(\mathcal{A}(\mathcal{X}_j), \omega). \tag{5}$$

Property 2. If any element of $\mathcal{X}_i$ is superior to any element of $\mathcal{X}_j$, and elements inside $\mathcal{X}_i$ or $\mathcal{X}_j$ are not ordered, the ranking loss can be expressed as the product of two summations over $\mathcal{X}_i$ and $\mathcal{X}_j$:

$$\mathcal{X}_i \prec_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j \Rightarrow R_e(\mathcal{X}_i \times \mathcal{X}_j, \omega) = \left(\sum_{x \in \mathcal{X}_i} e^{f_\omega(x)} \right) \left(\sum_{x' \in \mathcal{X}_j} e^{-f_\omega(x)} \right) \tag{6}$$

Reduction of the complexity. Using these properties, we will now propose a way for reducing the complexity of minimizing Function 4. It is based on

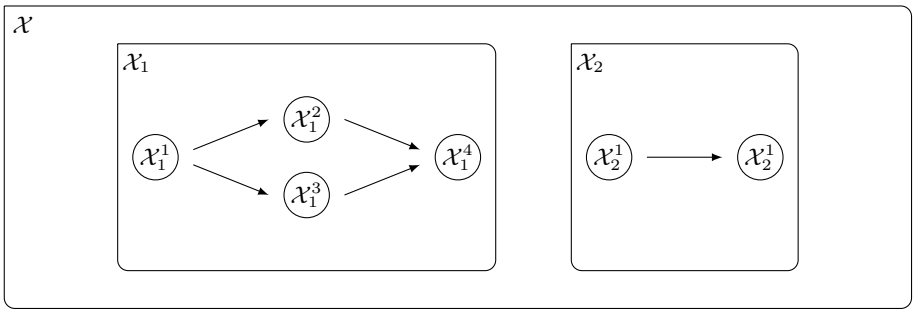

Fig. 1. Representation of a partition of $\mathcal{X}$. There is no order between an element of $\mathcal{X}_1$ and an element of $\mathcal{X}_2$. Inside $\mathcal{X}_1$, the arrow $\mathcal{X}_1^1 \rightarrow \mathcal{X}_1^2$ means that an element of $\mathcal{X}_1^1$ must be higher ranked than an element of $\mathcal{X}_1^2$. Inside any $\mathcal{X}_i^j$, there is no order.

a decomposition of $R_e(\mathcal{A}(\mathcal{X}), \omega)$ according to the greatest subsets of $\mathcal{X}$ which verify either of the conditions expressed in the left hand side of Equations 5 or 6 ($\mathcal{X}_i \perp_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j$ or $\mathcal{X}_i \prec_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j$). These subsets are denoted $\mathcal{X}_i^j$ in the following.

Let $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n$ be a partition of $\mathcal{X}$ such that:

$$\forall (i,j) \in \{1, ..., n\}^2 : \mathcal{X}_i \perp_{\mathcal{P}(\mathcal{X})} \mathcal{X}_j. \tag{7}$$

The subsets used for the decomposition of the error Function, $\mathcal{X}_k^1, \mathcal{X}_k^2, ..., \mathcal{X}_k^{n_k}$, will be a partition of $\mathcal{X}_k$ such that:

$$\forall (i,j) \in \{1, ..., n_k\}^2 : \begin{cases} \mathcal{X}_k^i \perp_{\mathcal{P}(\mathcal{X})} \mathcal{X}_k^j \text{ or} \\ \mathcal{X}_k^i \prec_{\mathcal{P}(\mathcal{X})} \mathcal{X}_k^j \text{ or} \\ \mathcal{X}_k^j \prec_{\mathcal{P}(\mathcal{X})} \mathcal{X}_k^i \end{cases} \tag{8}$$

An example of possible partition is represented in figure 1.

According to property 1 and property 2, the exponential loss (4) can be rewritten:

$$R_e(\mathcal{A}(\mathcal{X}), \omega) = \sum_{k=1}^{n} \sum_{i=1}^{n_k} \left\{ \left(\sum_{x \in \mathcal{X}_k^i} e^{f_\omega(x)} \right) \left(\sum_{\substack{j \in [1, n_k] \\ \mathcal{X}_k^j \prec \mathcal{X}_k^i}} \sum_{x' \in \mathcal{X}_k^j} e^{-f_\omega(x')} \right) \right\}. \tag{9}$$

The complexity for computing this expression is $O(n \cdot K \cdot |\mathcal{X}|)$ whereas it is $O(n \cdot |\mathcal{X}|^2)$ for Function 4 where K is the total number of subsets $\mathcal{X}_j^i$ in the partition of $\mathcal{X}$. The worst case occurs when $K = |\mathcal{X}|$.

Gradient descent. Since Function 9 is convex, we can use a gradient descent technique to minimize it. The components of the gradient has the following form:

$$\frac{\partial R_e(\mathcal{A}(\mathcal{X}))}{\partial \omega_p} = \sum_{k=1}^{n} \sum_{i=1}^{n_k} \left\{ \left(\sum_{x \in \mathcal{X}_k^i} t_p e^{f_\omega(x)} \right) \left(\sum_{\substack{j \in [1,n_k] \\ \mathcal{X}_k^j \prec \mathcal{X}_k^i}} \sum_{x' \in \mathcal{X}_k^j} e^{-f_\omega(x')} \right) + \left(\sum_{x \in \mathcal{X}_k^i} e^{f_\omega(x)} \right) \left(\sum_{\substack{j \in [1,n_k] \\ \mathcal{X}_k^j \prec \mathcal{X}_k^i}} \sum_{x' \in \mathcal{X}_k^j} -t'_p e^{-f_\omega(x')} \right) \right\} . \quad (10)$$

The complexity for computing the gradient is the same ($O(n \cdot K \cdot |\mathcal{X}|)$) as that of Function 9.

3 Application to Structured Information Retrieval

3.1 Definitions

Suppose we have a collection of hierarchically structured documents. Each document d can be represented by a tree T. Each node of the tree has a "type" (or tag) and a textual content. $\mathcal{L}$ will denote the set of node types.

For each node n of T, the doxel at node n is the subtree T_n of T rooted at n.

We use the following notations, $\mathcal{D}$ is the set of doxels for all the documents in the collection, $\mathcal{Q}$ is a a set of information needs and $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$ is the set of elements we want to order.

We suppose that there exists a partial order $\prec$ on $\mathcal{X} = \mathcal{Q} \times \mathcal{D}$, this partial order will reflect for some information needs, the evidence we have about preferences between doxels. It is provided via user feedback or manual assessments of the SIR corpus. Note that these relevance assessments are needed only on a subpart of the collection We consider here the task which consists in producing a ranked list of doxels which fulfill an information need $q \in \mathcal{Q}$. For that, we will train the ranking model to learn a total strict order on $\mathcal{X}$.[3]

3.2 Representation

Each element $x \in \mathcal{X}$ is represented by a vector $(t_1, t_2, ..., t_d)$ were t_i represents some feature which could be useful to order elements of $\mathcal{X}$.

To take into account the structural information, we will use the information provided by the context of the doxel and the information given by the node type of the doxel. For the former, we will use features characterizing the doxel, its parent, and the whole document. For the latter, label information can be used by ranking doxels with the same node type which leads to learn a $f_{\omega|l}$ for each node type l, and then using all $f_{\omega|l}(x)_{l=1..|\mathcal{L}|}$ as features in a second ranking step.

[3] For simplification, we consider here a total strict order on the doxels, and not the case where different doxels may have the same rank.

These two steps can be reduced to one using the following vector representation. If we denote $x^l = (t_0^l, t_1^l, t_2^l, ..., t_d^l)$, where $t_0^l = 1$ is a bias term, the vector representing x for node type l, we have $t_i^l = t_i$ if l is the node type of the root of x and $t_i^{(l)} = 0$ if not. We will consider the ranking of global vectors defined as follows:

$$x = \left((t_0^{l_1}, t_1^{l_1}, t_2^{l_1}, ..., t_d^{l_1}), (t_0^{l_2}, t_1^{l_2}, t_2^{l_2}, ..., t_d^{l_2}), ..., (t_0^{l_{|\mathcal{L}|}}, t_1^{l_{|\mathcal{L}|}}, t_2^{l_{|\mathcal{L}|}}, ..., t_d^{l_{|\mathcal{L}|}}) \right).$$

Where $|\mathcal{L}|$ is the number of different labels in the collection. In the above expression all vector components of the form $(t_i^{l_i}, t_i^{l_i}, t_2^{l_i}, ..., t_d^{l_i})$ will be equal to $(0, 0, ..., 0)$ except for one which corresponds to x label. This representation allows computing in one step the ranking of nodes from different types.

3.3 Reduction of Complexity

In order to reduce the complexity, we have to find the subsets $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n$ of $\mathcal{X}$ which verify the condition (7) in section 2.2. We can easily find such subsets, if we denote $(q_i)_{i=1..|\mathcal{Q}|}$ the elements of $\mathcal{Q}$, there are at least for each q_i:

$$\mathcal{X}_i = \{x = (d, q) \in \mathcal{X} / q = q_i\}$$

$\mathcal{X}_i$ is the set of couples (doxel, information need) which corresponds to the same information need. A corollary of this property is that it is useless to compare scores of doxels from different queries. For each $\mathcal{X}_i$'s, the preferences among doxels may be expressed according to several discrete dimensions. For example in INEX, we have:

- an information of exhaustivity, which measures how much a doxel answers the totality of an information need (0 not exhaustive, ..., 3 fully exhaustive)
- an information of specificity, which measures how much a doxel answers only the information need (0 not specific, ..., 3 means fully specific)

A doxel labeled E_3S_3 (which means fully exhaustive and specific) is greater than one labeled E_1S_3 (which means marginally exhaustive and fully specific).

If such a discrete multidimensional scale exists, we can find a partition according to (8) by considering for each $\mathcal{X}_i^j$ a set of doxels whose assessments share the same values along all dimensions. For instance, we can choose $\mathcal{X}_i^0$ to be the set of doxels assessed E_0S_0 for the information need q_i.

4 Experiments

4.1 Test Collection

To evaluate our method, we used the INEX document, topic and assessment collection. This collection contains 16819 XML documents representing the content of the articles of the IEEE Computer Society's journal publications from 1995 to 2004. These documents are represented in the same DTD. In the year 2003, 36 content-oriented topics with the corresponding assessments on doxels were produced. In 2004, 40 topics and assessments were added. The assessments for 2003

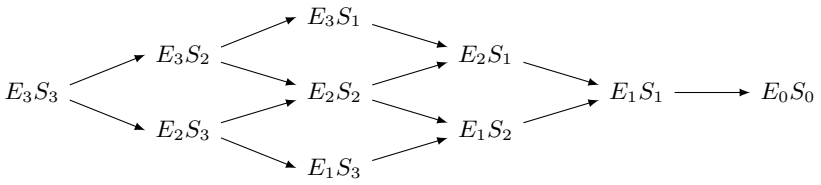

Fig. 2. Graph representing the order between elements for a given information need, according to the two dimensional discrete scale of INEX. Doxels labeled E_3S_3 must be the highest ranked, and doxels labeled E_0S_0 the lowest ranked.

and 2004 only concern 12107 documents from 1995 to 2002, since the rest of the collection was not available at this time. In 2005, the database has been extended with document from 2003 to 2004 and 40 topics and assessments were added.

A content-oriented topic is a list of words representing an information need.

An assessment is an evaluation of how much a particular doxel of the document collection answers the information need. In INEX, assessments are expressed in a two dimensional discrete scale which has been described above in section 3.3. The assessments and the trellis giving the partial ordering between these assessments are described in Figure 2.

4.2 Representation

For computing features, we used an Okapi model[4] [14], which is one of the reference models on flat documents. Okapi was adapted so as to reach good performance on the collection of structured documents. This adaptation consists in using doxels rather than documents for computing the term frequencies, and using as normalization factor for each doxel, the mean size of the doxels with the same node type.

In the experiments, we used three features for the combination:

- t_1 = the Okapi score of the doxel
- t_2 = the Okapi score of the parent of the doxel
- t_3 = the Okapi score of the whole document

these features provide some information about the structural context of a doxel. Sets of node types were defined according to the DTD of the document collection: article, abstract, sections, paragraphs, lists... Node type was introduced according to the method described in section 3.2.

We used the series of topics and assessments from the INEX 2003 and 2004 collections as a learning base and those from 2005 as a test base.

4.3 Filtering

For some SIR systems, returning overlapping doxels could be an undesirable behaviour, which means for example that it should not return a section, and

[4] With parameters $k_1 = 2.0$, $k_3 = 7.0$ and $b = 0.75$.

one of its paragraphs. In order to suppress all overlapping elements from an existing list, we used a strategy which consists in removing all elements which are overlapping with an element ranked higher in the list. This is the simplest way to remove overlap and this allows us to focus on the efficiency of the ranking algorithm rather than the filtering technique. Other ways of limiting overlapping can be found in [15]. Two kinds of experiences have been carried out : 2 without removing overlap, and 2 where overlap was removed.

4.4 Evaluation

We used two metrics to evaluate our approach:

- a precision recall metric which does not take into account overlapping elements;
- a cumulated gain based metric [16]. This metric, developed for the evaluation of INEX [17], considers the dependency of XML elements, and will penalize ranked lists with overlapping elements.

4.5 Results

With filtering. We have plotted in figures 3 and 4 the evaluation of the lists produced by the ranking algorithm and by the modified Okapi where overlap was removed. We can see for both metrics that the ranking algorithm performs better than Okapi. The difference for the precision/recall metric is not large: this is due to the post filtering of the lists. The ranked lists had not been optimized for non overlapping elements since there is no notion of overlapping in the exponential loss.

Without filtering. Figures 5 and 6 show the evaluation of the lists produced by the ranking algorithm and modified Okapi where overlap was removed. We can see for both metrics that the ranking algorithm performs clearly better than

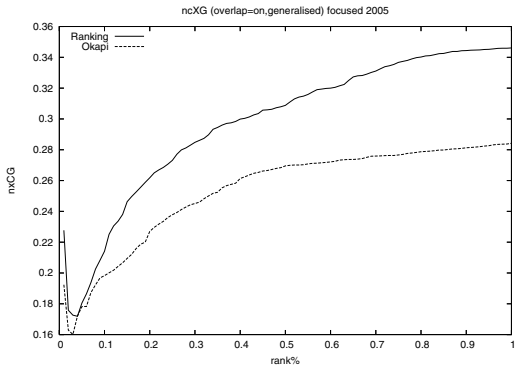

Fig. 3. Performance of Ranking and Okapi models with filtering evaluated with the cumulated based metric ncXG

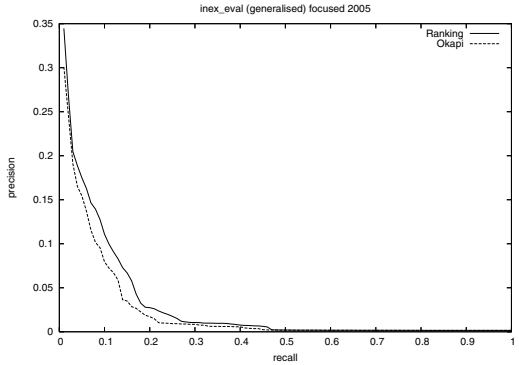

Fig. 4. Performance of Ranking and Okapi models with filtering evaluated with the precision/recall metric

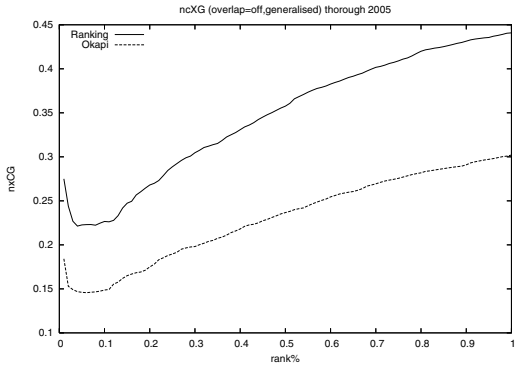

Fig. 5. Performance of Ranking and Okapi models without filtering evaluated with the cumulated based metric ncXG

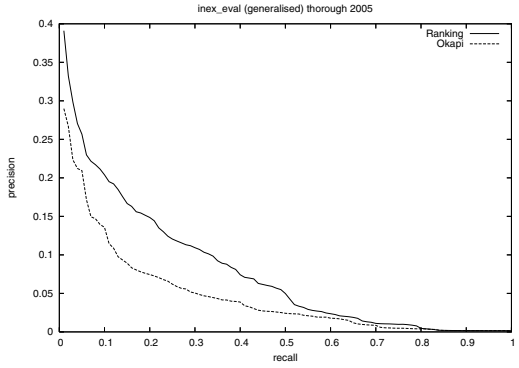

Fig. 6. Performance of Ranking and Okapi models without filtering evaluated with the precision/recall metric

Okapi and the difference in performance is superior than in the non overlapping case.

For both experiments, the ranking algorithm has been able to increase the performance of the baseline Okapi. Ranking methods thus appear as a promising direction for improving SIR search engine performance. It remains to perform tests with additional features (for example the scores of additional IR systems).

5 Conclusion

We have described a new model for performing Structured Information Retrieval. It relies on a combination of scores from the Okapi model and takes into account the document structure. This score combination is learned from a training set by a ranking algorithm. We have shown that learning to rank document elements improves a baseline model Okapi, which is known to be effective on IR on flat documents.

Acknowledgements

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. Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z., eds.: Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers. In Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z., eds.: INEX. Volume 3493 of Lecture Notes in Computer Science., Springer (2005)
2. Baeza-Yates, R., Maarek, Y.S., Roelleke, T., de Vries, A.P.: Third edition of the "XML and Information Retrieval" workshop. first workshop on integration of ir and db (wird) jointly held at sigir'2004, sheffield, uk, july 29th, 2004. SIGIR Forum (2004) 24–30
3. Lalmas, M.: Dempster-shafer's theory of evidence applied to structured documents: Modelling uncertainty (1997)
4. Lalmas, M., Moutogianni, E.: A dempster-shafer indexing for the focussed retrieval of hierarchically structured documents: Implementation and experiments on a web museum collection (2000) RIAO, Paris, France.
5. Piwowarski, B., Gallinari, P.: A bayesian network for XML Information Retrieval: Searching and learning with the INEX collection. Information Retrieval (2004)
6. Ogilvie P., Callan J.: Using Language Models for Flat Text Queries in XML Retrieval. In Proceedings of INEX 2003 (2004) 12–18
7. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: Advances in Neural Information Processing Systems. Volume 10., The MIT Press (1998)

8. Bartell, B.T., Cottrell, G.W., Belew, R.K.: Automatic combination of multiple ranked retrieval systems. In: Research and Development in Information Retrieval. (1994) 173–181
9. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. In Shavlik, J.W., ed.: Proceedings of ICML-98, 15th International Conference on Machine Learning, Madison, US, Morgan Kaufmann Publishers, San Francisco, US (1998) 170–178
10. Amini, M.R., Usunier, N., Gallinari, P.: Automatic text summarization based on word-clusters and ranking algorithms. In: ECIR. (2005) 142–156
11. Craswell, N., Robertson, S., Zaragoza, H., Taylor, M.: Relevance weighting for query independent evidence. In: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2005) 416–423
12. Auer, P., Meir, R., eds.: Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings. In Auer, P., Meir, R., eds.: COLT. Volume 3559 of Lecture Notes in Computer Science., Springer (2005)
13. Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In Belkin, N.J., Ingwersen, P., Pejtersen, A.M., eds.: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992, New York, ACM (1992) 198–210
14. Robertson, S.E., Walker, S., Hancock-Beaulieu, M., Gull, A., Lau, M.: Okapi at TREC. In: Text REtrieval Conference. (1992) 21–30
15. Kazai, G., Lalmas, M., Rölleke, T.: A model for the representation and focussed retrieval of structured documents based on fuzzy aggregation. In: SPIRE. (2001) 123–135
16. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. (2002) 422–446
17. Kazai, G., Lalmas, M.: Inex 2005 evaluation metrics. Technical document (2005)