Integrating structure in the probabilistic model for Information Retrieval

Mathias Géry, Christine Largeron, Franck Thollard

Integrating structure in the probabilistic model for Information Retrieval

Mathias Gery Christine Largeron Franck Thollard
Université de Lyon, UMR-5516 Saint-Etienne, France
{mathias.gery,largeron,thollard}@univ-st-etienne.fr

Abstract

*In databases or in the World Wide Web, many documents are in a structured format (*e.g. *XML). We propose in this article to extend the classical IR probabilistic model in order to take into account the structure through the weighting of tags. Our approach includes a learning step in which the weight of each tag is computed. This weight estimates the probability that the tag distinguishes the terms which are the most relevant. Our model has been evaluated on a large collection during INEX IR evaluation campaigns.*

1 Introduction

Most of the available information either in textual databases or on the Internet is strongly structured. This is for example the case for scientific articles or for the documents available on Internet when they are written in markup languages (e.g. HTML or XML). For all these documents, information provided by structure can be used to emphasize some particular words or some parts of the textual document. Consequently, a word does not have the same importance if it is emphasized (e.g. bold font, italic, etc.) or not. Similarly, a given word does not have the same importance depending on its position in the document: a term found in the title could have more importance than the one that appears in the text body. This is also true for the documents written in markup languages (e.g. HTML or XML) that explicitly describe the logical structure of the document (*e.g.* title, section, etc.).
Several models have been proposed to represent documents. They can be organized in three families [1] depending on the model on which they are based: boolean model, vector space model or probabilistic model. However all these models use flat text representations and consequently they do not take into account the structure of the documents.
For all these reasons, recent researches have focused on exploiting the structure of documents.

In the context of information retrieval (IR), taking into account the structure can be done either at the step of querying or at the step of indexing.

One way to integrate the structure at the step of querying is to adapt query languages like in [11, 14], XIRQ [5, 4] or NEXI [1]. The limits of these approaches come from their assumption: they assume the user knows where the most relevant information is located in documents and that he is able (and will want) to use complex queries; this is however not true in practice [8].

The second approach consist in integrating the structure at the indexing step by introducing a structure weighting scheme [6]. In such a scheme, the weight assigned to a word is not only based on its frequency in the document and possibly in the collection, but also on its position in the document. The tags are used to define these positions. Hence, the ranking is not function of the existence of a term in a document but of the existence of a term marked by the appropriate tag. Different tags can be considered: tags related to formatting (*e.g.* bold or italic markup) and logical tags that define an internal structure and that can be used to split the documents into elements (*e.g.* section, paragraph, etc.) or to represent it as a tree.

From a methodological point of view, this second approach raises two questions: how to choose the structural weights, and how to integrate them in the usual models ?

In the first works, the tags used to define the structure as well as their weight were chosen empirically [16]. More recent studies propose to learn automatically the weights assigned to tags, using optimizing techniques such as genetic algorithms [10, 22] or simulated annealing [2].

The second question concerns the way of integrating the structure weighting in the usual models. When only logical tags are considered, it is possible to split the document into parts (like title, abstract, body, etc.) defined by these tags and to process independently each part. Then a linear combination of the scores obtained on each part can be computed. But, the experiments, presented in [18], suggest that it could be more advantageous to duplicate each part according to its weight than to use a linear combination of the scores obtained on each part. However, it is worth stressing

[1] NEXI is the query language used in the INEX competition.

that both techniques are not appropriate to retrieve parts of documents (e.g. elements).

An alternative approach has been developed, based on the tree structure of XML documents [12, 15, 20, 22]. The XML elements (*i.e.* the nodes of the tree) are represented by paths and the position of a word on the path is considered. For example in [12], a term located at position `journal/issue/article/title` has a larger weight than a term located at `journal/issue/article/abstract`. In [20], the inverse document frequency for a term is computed for each subtree in the collection, while in [22] the structural weights associated to nodes are learned using genetic algorithm.

The main problem with these tree based approaches, is the number of parameters that need to be tuned since this number increases with the size and the heterogeneity of the collection; this is one limit for the application of such models on large collections available on the web and it is probably the reason for which in most of these articles, experiments are presented with few tags, most often less than five ones.

The approach adopted here consists in associating a weight to each tag instead of a weight to a sequence of tags like in previous articles based on tree representation. The weights of the tags are used as multiplicative factors of the term frequency weights of the word within their scope. This approach has already been used to improve the probabilistic model [24, 13] as well as the vector space model [23, 2], with a limited set of tags. In these works, tags weights need to be tuned and Lu et *al.* note that "the creation of a practical algorithm to generate values for tuning parameters at the element level is a challenging task". This is the aim of this work. Indeed, in this article [2], we propose a formal framework in order to take into account the document structure (either the structural tags or formatting tags) in the probabilistic model. Our approach is made up of two steps, the first one is a learning step, in which a weight is computed for each tag. This weight estimates the probability that a tag distinguishes terms which are the most relevant. In the second step, these tags weights are used to better estimate the relevance of a document for a given query. Beyond a scalar weighting scheme, this framework allows to compare several ways (called PSPM, CSPM and ASPM schemes) of combining tags weights and terms weights, including one scheme based on sequences of tags (PSPM scheme).

An overview of our model is given in section 2. A more formal presentation follows in section 3. Some results obtained on the INEX collections are presented in section 4.

[2]This work has been partly funded by the Web Intelligence project (région Rhône-Alpes).

2 Modelling document structure

In information retrieval, the probabilistic model [17] aims at estimating the relevance of a document for a given query through two probabilities: the probability of finding a relevant information and the probability of finding a non relevant information. These estimates are based on the probability that a given term appears in relevant (or in non relevant) documents. Given a training collection in which the documents relevance for some queries is available, one can estimate the probability that a given term belongs to a relevant document (respectively to a non relevant document), given its distribution in relevant documents (respectively in non relevant ones).

Our goal here is to extend the probabilistic model by taking into account the documents structure. In our model, we consider two kinds of XML tags, according to their role: the logical tags (title, section, paragraph, etc.) and the formatting ones (bold font, centered text, etc.). The logical tags are used in order to select the XML elements (section, paragraph, table, etc.) that are handled at the indexing step. These indexed elements are those which then can be ranked and returned to the user. The formatting tags are integrated in the probabilistic model in order to improve the terms weighting. Comparatively to the probabilistic model, the tag weight aims at distinguishing relevant terms. But, in our approach, the tags are considered instead of terms and the terms are used instead of documents. The relevance is therefore evaluated on parts of documents (term by term) instead of whole documents, and the probability of a tag to distinguish the relevant terms is estimated on the elements (i.e. XML elements). At the step of querying, the probability for an element to be relevant is estimated not only on the weights of terms it contains, but also on the weights of the tags which label these terms.

A more formal presentation of our model is given in the next section.

3 A probabilistic model for the representation of structured documents

3.1 Notations and examples

Let $\mathcal{D}$ be a set of structured documents. Without loose of generality, we will consider here XML documents. Each tag describing logical structure (*article*, *section*, *p*, etc.) defines elements that correspond to parts of document. Therefore, each logical element (article, section, paragraph, table, etc.) will be represented by a set of terms and will be indexed.

We note:

- $E = \{e_j, j = 1, ..., l\}$, the set of the logical elements available in the collection (*article*, *section*, *p*, etc.);

- $T = \{t_1, ..., t_i, ...t_n\}$, a term index built from E;
- $B = \{b_1, ..., b_k, ..., b_m\}$, the set of tags.

In the following section, the representation of an element e_j is noted x_j when only the terms are considered, and m_j when both terms and tags are taken into account.

3.2 Term based score for an XML element to be relevant

In the probabilistic model, the relevance of an element, for a given query Q, is function of the weights of the matching terms (*i.e.* terms of the query contained in the element). In order to evaluate this relevance, a ranking function computes a term weighting score. In this function, the weight of term t_i in element x_j is noted w_{ji}. We suppose that the weight of term t_i in Q equals 1.

Formally, we define X_j a vector of random variables and $x_j = (x_{j1}, ..., x_{ji}, ..., x_{jn})$ a realization of the vector X_j, with $x_{ji} = 1$ (resp. 0) if terms t_i appears (resp. does not appear) in element e_j. Given these notations, the term based relevance f_{term} of x_j is given by the score:

$$f_{term}(x_j) = \sum_{t_i \in T \cap Q} x_{ji} \times w_{ji} \quad (1)$$

This general dot-product form covers different ranking functions, for example the functions `ltn` and `ltc` implemented by the SMART system [19], or the well known `BM25` function [17](cf. equation 2) used during our experiments (cf. section 4):

$$w_{ji} = \frac{tf_{ji} * (k_1 + 1)}{k_1 * ((1 - b) + (b * ndl)) + tf_{ji}} * \log \frac{N - df_i + 0.5}{df_i + 0.5} \quad (2)$$

with:

- tf_{ji}: the frequency of t_i in element e_j.
- N: the number of all elements in the collection.
- df_i: the number of elements containing the term t_i.
- ndl: the ratio between the length of element e_j and the average element length.
- k_1 and b: the classical BM25 parameters (tuned to 1.1 and 0.75).

Given this classical term base model, the goal is now to propose an extension that will take into account the documents structure.

3.3 Tag based score for an XML element to be relevant

Similarly to the previous section, we define M_j as a vector of random variables T_{ik} in $\{0, 1\}$:

$$M_j = (T_{10}, ..., T_{1k}, ..., T_{1m}, ..., T_{n0}, ..., T_{nk}, .., T_{nm})$$

with
$T_{ik} = 1$ if term t_i appears in this element labeled by b_k
$T_{ik} = 0$ if term t_i does not appear labeled by b_k
$T_{i0} = 1$ if term t_i appears without being labeled by a tag in B
$T_{i0} = 0$ if term t_i does not appear without being labeled

We note
$m_j = (t_{10}, ..., t_{1k}, ..., t_{1m}, ..., t_{i0}, ..., t_{ik}, .., t_{im}, ..., t_{n0}, ..., t_{nk}, .., t_{nm})$
a realization of the random variable M_j.

We want to estimate the relevance of an XML element e_j (modeled by the vector m_j) for a given query :

- $P(R|m_j)$: the probability of finding a relevant information (R) given an element m_j and a query.
- $P(NR|m_j)$: the probability of finding a non relevant information (NR) given an element m_j and a query.

Let $f_1(m_j)$ be a document ranking function:

$$f_1(m_j) = \frac{P(R|m_j)}{P(NR|m_j)}$$

The higher $f_1(m_j)$, the more relevant the information presented in m_j. Using Bayes formula, we get:

$$f_1(m_j) = \frac{P(m_j|R) \times P(R)}{P(m_j|NR) \times P(NR)}$$

The term $\frac{P(R)}{P(NR)}$ being constant over the collection for a given query, it will not change the ranking of the documents. We therefore define f_2 (which is proportional to f_1) as:

$$f_2(m_j) = \frac{P(m_j|R)}{P(m_j|NR)}$$

Using the Binary Independence Model assumption, we have:

$$P(M_j = m_j|R) = \prod_{t_{ik} \in m_j} P(T_{ik} = 1|R)^{t_{ik}} P(T_{ik} = 0|R)^{1 - t_{ik}} \quad (3)$$

In the same way, we get :

$$P(M_j = m_j|NR) = \prod_{t_{ik} \in m_j} P(T_{ik} = 1|NR)^{t_{ik}} P(T_{ik} = 0|NR)^{1 - t_{ik}} \quad (4)$$

For sake of notation simplification, we note, for a given XML element:

$p_0 = P(T_{i0} = 0|R)$**:** the probability that t_i does not appear without being labeled, given a relevant element.

$p_{ik} = P(T_{ik} = 1|R)$**:** the probability that t_i appears marked by the tag k, given a relevant element.

$q_0 = P(T_{i0} = 0|NR)$**:** the probability that t_i does not appear without being labeled, given a non relevant element.

$q_{ik} = P(T_{ik} = 1|NR)$**:** the probability that t_i appears marked by the tag k, given a non relevant element.

Using these notations in equations 3 and 4, we get:

$$P(m_j|R) = \prod_{t_{ik} \in m_j} (p_{ik})^{t_{ik}} \times (1 - p_{ik})^{1-t_{ik}},$$

$$P(m_j|NR) = \prod_{t_{ik} \in m_j} (q_{ik})^{t_{ik}} \times (1 - q_{ik})^{1-t_{ik}}.$$

The ranking function $f_2(m_j)$ can then be re-written:

$$f_2(m_j) = \frac{\prod_{t_{ik} \in m_j} (p_{ik})^{t_{ik}} \times (1 - p_{ik})^{1-t_{ik}}}{\prod_{t_{ik} \in m_j} (q_{ik})^{t_{ik}} \times (1 - q_{ik})^{1-t_{ik}}}$$

The *log* function being monotone increasing, taking the logarithm of $f_2$will not change the ranking:

$$\begin{aligned} f_3(m_j) &= \log(f_2(m_j)) \\ &= \sum_{t_{ik} \in m_j} t_{ik} \times \left(\log\left(\frac{p_{ik}}{1 - p_{ik}}\right) - \log(\frac{q_{ik}}{1 - q_{ik}}) \right) \\ &\quad + \sum_{t_{ik} \in m_j} \log(\frac{1 - p_{ik}}{1 - q_{ik}}) \end{aligned}$$

As before, the term $\sum_{t_{ik} \in m_j} \log(\frac{1-p_{ik}}{1-q_{ik}})$ is constant with respect to the collection (independent of t_{ik}). Not considering it will not change the ranking provided by $f_3(m_j)$:

$$f_{tag}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} t_{ik} \times \log\left(\frac{p_{ik}(1 - q_{ik})}{q_{ik}(1 - p_{ik})}\right) \quad (5)$$

Thus, we obtain in this ranking function, a weight w'_{ik} for each term t_i and each tag b_k:

$$w'_{ik} = \log(\frac{p_{ik}(1 - q_{ik})}{q_{ik}(1 - p_{ik})}) \quad (6)$$

Finally, in our probabilistic model that takes into account the document structure, the relevance of an XML element m_j, relatively to the tags, is defined through $f_{tag}(m_j)$:

$$f_{tag}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} t_{ik} \times w'_{ik} \quad (7)$$

This formula is similar to the classical term weighting function seen in equation 1, except that tags weights are considered instead of terms weights.

3.4 Estimation of tags weights

In practice, we have to estimate the probabilities p_{ik} and q_{ik}, $i \in \{1, .., n\}$, $k \in \{0, .., m\}$ in order to evaluate the relevance of an element. For that purpose, we used a learning set LS in which the relevance of the elements for a given query is known [3] Given the set R (respectively NR) that contains the relevant elements (respectively non relevant ones) a contingency table can be built for each term t_i labeled by b_k:

	R	NR	$LS = R \cup NR$
$t_{ik} \in m_j$	r_{ik}	$nr_{ik} = n_{ik} - r_{ik}$	n_{ik}
$t_{ik} \notin m_j$	$R - r_{ik}$	$N - n_{ik} - R + r_{ik}$	$N - n_{ik}$
Total	R	$\|NR\| = N - R$	N

with:

- r_{ik}: the number of times term t_i labeled by b_k is relevant in LS;
- $\sum_i r_{ik}$: the number of relevant terms labeled by b_k in LS.
- n_{ik}: the number of times term t_i is labeled by b_k in LS;
- $nr_{ik} = n_{ik} - r_{ik}$: the number of times term t_i labeled by b_k is not relevant in LS;
- $R = \sum_{ik} r_{ik}$: the number of relevant terms in LS;
- $|NR| = N - R$: the number of non relevant terms in LS.

We can now estimate $\begin{cases} p_{ik} = & P(t_{ik} = 1|R) & = \frac{r_{ik}}{R} \\ q_{ik} = & P(t_{ik} = 1|NR) & = \frac{n_{ik} - r_{ik}}{N - R} \end{cases}$

And w'_{ik} follows:

$$w'_{ik} = \log \frac{r_{ik} \times (|NR| - nr_{ik})}{nr_{ik} \times (R - r_{ik})} \quad (8)$$

This weighting function evaluates the probability, for a given tag, to distinguish relevant terms from non relevant ones. This is closely related to the probabilistic model, in which a weight is estimated for each term, based on the probability that this term appears in relevants documents or in non relevants documents (cf. section 3.2). But in our approach, the tags are considered instead of terms and the terms are used instead of documents.

[3] For instance, in our experiments, we use the Inex 2006 assessments as training set: the parts of text, noted by experts as relevant or not relevant, are available for 114 queries. We used these assessments for building the contingency table.

For pratical purpose, we propose to estimate a global weight w'_k for each tag b_k instead of a weight for each couple (term, tag):

$$w'_k = \frac{\sum_{t_i \in T} w'_{ik}}{|T|}$$

Finaly, the tag based score will be:

$$f_{tag}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} t_{ik} \times \frac{\sum_{t_i \in T} w'_{ik}}{|T|} \quad (9)$$

3.5 Combining term based and tag based scores

In order to estimate the relevance of an element e_j given a query, a global ranking function $fc(e_j)$ combining terms weights used in $f_{term}(x_j)$ (eq. 1) and tags weights used in $f_{tag}(m_j)$ (eq. 9), is introduced:

$$fc(e_j) = \sum_{t_{ik} \in m_j / t_i \in Q} w_{ji} \times C_k(w'_k) \quad (10)$$

where C is a function that combines terms weights and tags weights.
We experimented different ways of combining terms weights and tags weights, in other words several functions C.

In the first one, called PSPM, the weight w_{ji} of each term t_i in m_j is multiplied by the weights w'_k of the tags b_k that mark this term. More formally:

$$f_{PSPM}(e_j) = \sum_{t_{ik} \in m_j / t_i \in Q} w_{ji} \times \prod_{k / t_{ik}=1} w'_k$$

We can note that some tags are going to reinforce the weight of the term ($w'_k > 1$) while other will weaken it ($w'_k < 1$).

The second model, called CSPM (for Closest Structured Probabilistic Model), only considers the weight w'_c of the tag b_c that tags the term t_i and that is the closest to t_i.

$$f_{CSPM}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} w_{ji} \times w'_c$$

In the third model, called ASPM (for Average Structured Probabilistic Model) the weight w_{ij} of each term t_i in m_j is multiplied with the average of the weights w'_k of the tags that label this term.

$$f_{ASPM}(m_j) = \sum_{t_{ik} \in m_j / t_i \in Q} w_{ij} \times \frac{\sum_{k / t_{ik}=1} w'_k}{|\{k / t_{ik} = 1\}|}$$

These strategies have been evaluated on the INEX 2006 & 2007 collections.

4 Experiments on INEX 2006 & 2007 collection - Track Adhoc

4.1 Experimental protocol

In our experiments, we used the english Wikipedia XML corpus [3] used during the INEX IR campaign (Initiative for Evaluation of XML Retrieval) 2006 & 2007 [7].

The corpus contains 659,388 articles extracted from the Wikipedia encyclopaedia in early 2006. The original Wiki syntax has been converted into XML, using both general tags of the logical structure (article, section, paragraph, title, list and item), formatting tags (like bold, emphatic) and frequently occurring link-tags. The documents are strongly structured as they are composed of 52 millions of XML elements. Each XML article can be viewed as a tree which contains, on average, 79 elements for an average depth of 6.72. Moreover, the whole collection (textual content + XML structure) represents 4.5 Gb while the textual content only 1.6 Gb. The structural information thus represents more than twice the textual one. In order to evaluate information retrieval systems, the INEX campaign has made available the relevance assessments corresponding to 114 queries in 2006, and to 107 queries in 2007. The corpus enriched by the INEX 2006 assessments has been used as the training set (LS) in order to build the contingency table described in section 3.4 and to estimate the tags weights w'_k. Then we have evaluated our approach using the 107 queries of INEX 2007.

We use the BM25 weighting function as the baseline, without stemming nor using a stoplist. In order to understand the pro and cons of our structured model, the same weighting function is used before integrating the tags weights in equation 10.

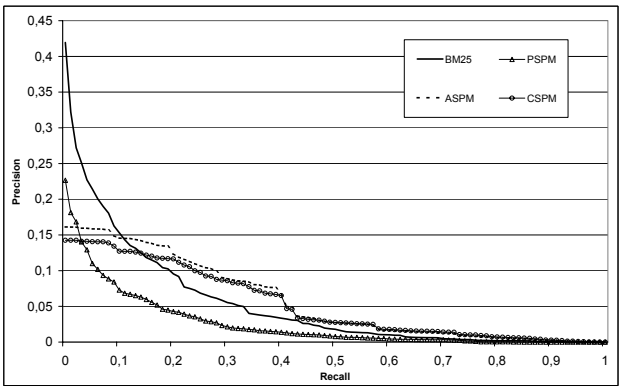
The evaluation measures are based on the *precision* and *recall* measures defined by [21]. The *interpolated average precision* (AiP), combines *precision* and *recall*, and provides an evaluation of the system results for each query. By averaging the AiP values on the set of queries, an overall measure of performance is defined in [9] and called *interpolated mean average precision* (MAiP).

4.2 Results

We have selected manually 14 tags in order to index the elements that will be returned to the user. These logical tags are: article, numberlist, body, title, p, row, section, td, table, tr, normallist, caption, th, definitionitem.

Regarding the other tags (namely the formatting tags), we first selected the 61 tags that appear more than 300 times in the 659,388 documents (collectionlink, item, unknownlink, cell, p, emph2, template, section, title, emph3, etc.).

Figure 1. MAiP of the three models evaluated on the 2007 collection



We then manually removed 6 of them, considered as not relevant (*e.g.* br, hr, value, ...). The weights of the 55 remaining tags were computed according to equation 8.

We now compare the results obtained on the 107 queries of INEX 2007 collection using our baselines and the three variants of our structured model defined in section 3.5: ASPM, PSPM and CSPM. BM25 and PSPM were only submitted as official runs to INEX 2007, but all the results were computed using the INEX evaluation programs (version 2, february 2008).

The results are synthesized either in table 1 and in figure 1. As can be seen, the BM25 baseline obtains a 5.32% MAiP, while PSPM obtains a 2.63% MAiP, CSPM 5.29% MAiP and ASPM 5.77% MAiP. The baseline is outperformed by our model ASPM, but produces better results than PSPM. Our interpretation of the latter is that multiplication has a too strong impact on small weights: two or three tags, which have small weight, are sufficient to delete a term. ASPM, that takes into account all tags by averaging their weights, performs slightly better than BM25 baseline. We can also notice that ASPM performs also better than CSPM (method that takes into account the closest tag). This is important since that shows that (some) tags can have a long term impact.

Our model is outperformed by the BM25 baseline at low recall levels (between P@0.00 and P@0.01), but it outperforms the BM25 baseline at other recall levels. In particular, we can see in table 1 that our model gives better results for P@0.90 and P@1.00. The precision of our model at P@1.00 is very low, but greater than zero. Each run submitted to INEX is a ranked list containing at most 1500 XML elements. This means that our model is sometimes able to retrieve all the relevant elements among the first 1500 XML returned elements.

Table 2. R[1500] and nMAiP of the three models evaluated on the 2007 collection

Model	R[1500]	nMAiP
BM25 (baseline)	0.1778	0.0095
PSPM: all tags (weights product)	0.1255	0.0033
CSPM: closest tags only	**0.7026**	0.0372
ASPM: all tags (average weights)	0.6951	**0.0401**

In order to better consider this fact in recall/precision curves, we have estimated $R[1500]$, the recall at 1500 elements, and we have used this value to normalize the recall-precision scores :

$$nMAiP = R[1500] * MAiP$$

These results show that our model outperforms the baseline (4.01% nMAiP versus 0.95% nMAiP), when exhaustiveness (i.e. $R[1500]$) is considered (cf. table 2).

The parameters of both models (namely BM25 and ASPM) were then tuned in order to understand the pros and cons of our approach. When accordingly tuned, BM25 reaches an 11.56% MAiP while ASPM obtains 9.52% MAiP. Further works are nevertheless needed in order to confirm this result, in particular regarding the combining scheme of terms and tags weights.

5 Conclusion

We proposed in this article a new way of integrating the XML structure in the classic probabilistic model. We consider both the logical structure and the formatting structure. The logical structure is used at indexing step for defining elements that correspond to part of documents. These elements will be indexed and potentially returned to the user. The formatting structure is integrated in the document model itself. During a learning step using the INEX 2006 collection, a weight is computed for each formatting tag, based on the probability that it distinguishes relevant terms. At the step of querying, the relevance of an element is evaluated using the weights of the terms it contains, but each term weight is modified by the weights of the tags that mark the term.

This model was evaluated on the INEX 2007 collection. In the experiments, the structured model outperforms the baseline. The different ways of combining terms weights and tags weights suggest a long term dependency between

Table 1. MAiP of the three models evaluated on the 2007 collection

Model	$iP[0.00]$	$iP[0.01]$	$iP[0.05]$	$iP[0.10]$	$iP[0.90]$	$iP[1.00]$	MAiP	Rank
BM25 (baseline)	**0.4195**	**0.3221**	**0.2142**	**0.1530**	0.0004	0.0000	0.0532	63th
PSPM: all tags (weights product)	0.2266	0.1813	0.1100	0.0729	0.0000	0.0000	0.0263	72th
CSPM: closest tags only	0.1426	0.1426	0.1405	0.1271	0.0027	0.0000	0.0529	
ASPM: all tags (average weights)	0.1611	0.1611	0.1584	0.1455	**0.0027**	**0.00001**	**0.0577**	

tags, as ASPM (Average Structured Probabilitic Model) outperforms a model in which only the closest tag is taken into account (namely CSPM). Moreover, it appears that the tags permit to improve the performance at high recall levels.

In a near future, we plan to analyze and take advantage of contextual information (*e.g.* long term dependency, relationship of the tag with respect to other tags, etc.). This can be done either from a practical point of view (*e.g.* using machine learning methods for modelling these relationships) or from theoretical point of view (*e.g.* by adapting the aggregation between term and tag weights in the structured probabilistic model).

Further work is nevertheless needed since the analysis of the size of the elements returned suggest that the information is more concentrated when dealing with elements rather than articles. To be more accurate, the sum of the size of the returned documents is around 17 millions of words in case of elements, and around 75 millions in case of articles. Since all the evaluations were made with the same bound on the number of elements to return (namely 1500), this means that the relevant information is much concentrated in the element returned that in the article ones.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999.

[2] J. Boyan, D. Freitag, and T. Joachims. A machine learning architecture for optimizing web search engines. In *AAAI Workshop on Internet-Based Information Systems*, 1996.

[3] L. Denoyer and P. Gallinari. The wikipedia XML corpus. In *SIGIR forum*, volume 40, pages 64–69, 2006.

[4] N. Fuhr and K. Grosjohann. XIRQL: A query language for information retrieval in XML documents. In *Research and Development in Information Retrieval*, 2001.

[5] N. Fuhr and K. Großjohann. XIRQL: An extension of XQL for information retrieval, July 2000. In ACM SIGIR, Workshop On XML and Information Retrieval, Athens, Greece.

[6] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Coherent answers for a large structured document collection. In *SIGIR*, pages 204–213, 1993.

[7] M. Géry, C. Largeron, and F. Thollard. UJM at INEX 2007: document model integrating XML tags. In *Proc. of INitiative for the Evaluation of XML Retrieval, Dagstuhl*, 2008.

[8] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Structured queries in XML retrieval. In *CIKM*, 2005.

[9] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In N. Fuhr, M. Lalmas, A. Trotman, and J. Kamps, editors, *Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval*, December 2007.

[10] Y.-H. Kim, S. Kim, J.-H. Eom, and B.-T. Zhang. Scai experiments on trec-9. In *Proc. of the Text Retrieval Conference (TREC-9)*, pages 392–399, 2000.

[11] D. Konopnicki and O. Schmueli. W3qs : A query system for the world-wide web. In *21th Intl. Conf. on Very Large Data Bases (VLDB'95)*, pages 54–65, Sept. 1995.

[12] E. Kotsakis. Structured information retrieval in XML documents, 2002. in 17th ACM Symposium on Applied Computing, pp 663-667, 2002.

[13] W. Lu, S. E. Robertson, and A. MacFarlane. Field-weighted XML retrieval based on bm25. In *INEX*, 2005.

[14] G. Navarro and R. A. Baeza-Yates. A language for queries on structure and contents of textual. In *SIGIR*, 1995.

[15] K. Pinel-Sauvagnat, L. Hlaoua, and M. Boughanem. XFIRM at INEX 2005: adhoc and relevance feedback tracks. In *INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl*, pages 88–103, 2005.

[16] J. Rapela. Automatically combining ranking heuristics for html documents. In *3rd International Workshop on Web Information and Data Management (WIDM) In Conjunction with ACM CIKM*, pages 61–67, 2001.

[17] S. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*, 27(3):129–146, 1976.

[18] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04*, pages 42–49, New York, NY, USA, 2004.

[19] G. Salton and M. McGill. *Introduction to modern Information Retrieval*. McGraw-Hill, 1983.

[20] T. Schlieder and H. Meuss. Querying and ranking XML documents. *JASIST journal*, 53(6):489–503, 2002.

[21] J. Swets. Information retrieval systems. *Science*, 141:245–250, 1963.

[22] A. Trotman. Choosing document structure weights. *Information Processing and Management*, 41(2):243–264, 2005.

[23] R. Wilkinson. Effective retrieval of structured documents. In *Research and Development in Information Retrieval*, 1994.

[24] J. E. Wolff, H. Florke, and A. B. Cremers. Searching and browsing collections of structural information. In *Advances in Digital Libraries*, pages 141–150, 2000.