Vojkan Mihajlović
Henk Ernst Blok
Djoerd Hiemstra
Peter M. G. Apers

Score Region Algebra: Building a Transparent XML-IR Database

Centre for Telematics and Information Technology (CTIT)
Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS)
University of Twente

Contents

Score Region Algebra: Building a Transparent XML-IR Database*

Vojkan Mihajlović Henk Ernst Blok Djoerd Hiemstra Peter M. G. Apers
CTIT, University of Twente
P.O. Box 217, 7500AE Enschede, The Netherlands
{v.mihajlovic, d.hiemstra, h.e.blok, p.m.g.apers}@utwente.nl

Abstract

A unified database framework that will enable better comprehension of ranked XML retrieval is still a challenge in the XML database field. We propose a logical algebra, named score region algebra, that enables transparent specification of information retrieval (IR) models for XML databases. The transparency is achieved by a possibility to instantiate various retrieval models, using abstract score functions within algebra operators, while logical query plan and operator definitions remain unchanged. Our algebra operators model three important aspects of XML retrieval: element relevance score computation, element score propagation, and element score combination. To illustrate the usefulness of our algebra we instantiate four different, well known IR scoring models, and combine them with different score propagation and combination functions. We implemented the algebra operators in a prototype system on top of a low-level database kernel. The evaluation of the system is performed on a collection of IEEE articles in XML format provided by INEX. We argue that state of the art XML IR models can be transparently implemented using our score region algebra framework on top of any low-level physical database engine or existing RDBMS, allowing a more systematic investigation of retrieval model behavior.

Keywords: information retrieval, structured documents, XML, region algebra, databases

1 Introduction

XML was initially developed as a standard for storing, carrying and exchanging data. With the rapid growth of data stored in XML format, ranked information retrieval (IR) from XML collections became vital requirement. Many systems have been developed in recent years that address this requirement.

1.1 From flat file to XML IR

An important class of XML IR systems is based on traditional (flat file) information retrieval methods (e.g., [24, 27, 34]) that represent a document as a "bag of words" [25] and where in most cases inverted file structures provide the basis for implementing a retrieval system. Although these systems are faster and simpler than any DBMS, they have important drawbacks when XML IR is being considered. Out of many properties of traditional IR systems, we discuss only the following four:

*This report is an extended version of the paper [31] published at Conference on Information and Knowledge Management (CIKM), Bremen, Germany, 31^{st} October - 5^{th} November, 2005.

- traditional IR systems lack the notion of *data independence* [14]: any change in what constitutes a document, or any change in document structure or used retrieval model, will lead to system developers needing to change major parts of the retrieval system
- traditional IR systems are developed for a simple query language, consisting in most cases of a set of terms
- most traditional IR systems are retrieval model specific, i.e., the retrieval model (or a small class of similar retrieval models) is hard-coded in the system
- traditional IR systems are developed for a specific storage structure, in most cases inverted files.

Although the concept of data independence was not a too big hurdle for the development of flat-file IR [21, 42], we argue that it is important for the development of XML IR. Unlike in the vast majority of flat file IR systems, where *documents* were the only units in which the user would search for information or which user would obtain as answer from the system, in XML IR the main focus is on nested XML *elements*. Furthermore, the notion of document is blurred as the whole collection of XML documents can be considered as one huge XML document with an artificial root element.

In XML IR, the user can specify not only his information need, but also where to search for information. This leads to the introduction of the specification of search elements in query languages used for the ranked retrieval on XML. Most of the XML IR query languages use the W3C query languages (XQuery [3] or XPath [8]) as a starting point and extend them with IR-like search expressions. Typical examples are full-text search extension of XQuery [1], Narrowed Extended XPath (NEXI) [39], and an extension of XQL, one of the predecessors of XQuery, named XIRQL [15].

Furthermore, hierarchical organization of XML documents enables the distinction between *search elements* and *answer elements* in XML IR, where both element types are not predefined as in flat file IR (documents). Search elements are elements where the user searches for specific information, while answer elements are elements that the user would like to obtain as an answer to a query. We illustrate an information retrieval search over XML documents for the user information need: "I would like to find sections addressing language models in an article that has an Abstract discussing information retrieval or a probabilistic database". This information need can be expressed in NEXI [39] (which uses a subset of XPath and extends XPaths with an *about* function for ranked retrieval and is adopted as an official query language in the INitiative for the Evaluation of XML Retrieval (INEX) [17]) as:

```
//article[about(.//abs, information retrieval)
   or about(.//abs, probabilistic database)]//sec[about(., language model)]
```

The example above introduces several additional query capabilities that are not clearly recognized in flat file retrieval:

- The search for information can be performed in arbitrary part of the XML document (or collection of XML documents) denoted with tags (search elements). In our example, search elements are 'article', 'abs' and 'sec'.
- The search element is not necessarily an answer element at the same time. Only 'sec' is a search and answer element at the same time in the example query (denoted with '.').
- The user can perform searches in different XML elements and later combine them to get the answer element. In our example information search is performed in 'abs' (abstract)[1] elements using the terms "information retrieval" and "probabilistic database" which are combined in an *OR* expression.

[1]Here we consider 'abs' as a strict condition although it can be considered as a hint for the retrieval system. See, e.g., vague content-and-structure (VCAS) queries in INEX [17].

The central part of any retrieval system is the retrieval model used. Although many approaches exist for XML information retrieval, most of the XML IR approaches are based on flat file tf.idf-like approaches [17]. For XML IR, the retrieval model has to incorporate additional aspects. We identify three aspects that a flexible XML retrieval system should provide to support XML-IR queries like the one above:

- element relevance score computation,
- element score combination, and
- element score propagation.

We use the term element *score* to denote the value that describes the estimated relevance of an element.

The first two aspects are inherited from the flat file IR model, although the second aspect can also denote the combination of scores for different search elements (see Section 2.1). The third one, sometimes called "augmentation" is XML specific and is recognized in the work of Fuhr and Großjohann [15] and Grabs and Shek [20], where the authors specified weighting factors for upwards propagation of scores in an XML tree representation. However, we think that the concept of downwards propagation should also be considered as, e.g., in our example we have to propagate relevance scores from the 'article' search element to the 'sec' answer element.

1.2 Towards the database approach

Besides numerous query languages, different physical implementation for ranked retrieval on XML have been proposed, ranging from the modified inverted file structures [19, 34] to full database implementations [13, 18]. For each indexing structure chosen for physical implementation, special algorithms need to be derived to enable fast execution of XML IR primitives. Whatever the primitives are, these algorithms will be dependent on XML storage structure.

Therefore, besides the adaptation of traditional IR systems, we can identify the other class of XML IR systems based on relational database technology. XML-IR database systems use well established database operators and thirty years' experience in RDBMS. RDBMS can be either enriched with an IR-like front end (loosely-coupled) [35] or tightly-coupled with IR search primitives [13, 15, 16], as recognized in [41]. Although relational systems are prevalent in the manipulation of documents structured as relations, they have difficulty handling nested structures such as XML. This is especially the case with handling containment relations (i.e., containment joins [29]), which are one of the basic operations used in information retrieval. That is why most of XML-IR database systems have so far been loosely-coupled systems. Recently, numerous enhancement have been proposed for handling containment relations (e.g., staircase join [22, 23], multipredicate merge join [43]), proving that relational technology can handle efficiently queries over XML data, including the containment queries.

The main characteristic of the database approach is a strong separation between conceptual, logical and physical levels [40]. By using different data models at each of those levels, *data abstraction* is provided. For XML-IR systems, following this separation in levels gives another, additional advantage over flat file IR systems: by choosing the appropriate level of abstraction for each level, the development of scoring techniques, handling structural information, is simplified, and kept *transparent* for the rest of the system design, making it flexible with respect to the query language and physical implementation used. Furthermore, the reasoning that can be made at the logical level can be useful for query rewriting and *optimization*. Using knowledge about the size of the operands and the cost for the execution of different operators at the physical level we are able to generate different logical query plans, speeding up the execution and lowering the memory requirements for query execution at the physical level.

Therefore, we identified the logical level as the central level that should provide the transparency considering XML-IR database systems. Unlike in the approach of Amer-Yahia et al. [1],

we want to integrate relevance score computations within the algebra. In [1] authors aimed to support a full-text search extension to XQuery (based on full-text query specification [6]), assuming that the scoring method is retrieval model implementation dependent, abstracting in that way from the problems of the XML IR and database integration and ones of element score propagation and combination. Therefore, our approach is closer to the approach of Fuhr et al. [15, 16], where the authors developed an XML IR path-based algebra and an XML IR query language named XIRQL. However, we based our algebra on containment relations among XML elements and not on the paths to XML elements. For the specification of our algebra we have chosen region algebra approaches [2, 5, 9, 37], because of their good properties in handling structured documents such as XML.

The basic idea behind the region algebra approaches is the representation of text documents as a set of *regions* (originally called extents), where each region is defined by its start and end positions. The aim of the region algebra approaches is modeling search in (semi-)structured documents using containment and set operators, The earliest region algebra approaches were a PAT system presented in [37], and the work of Burkowski [5] and Clarke et al. [9] in the area of textual databases. These approaches were later extended to support new operators, such as positional inclusion and direct inclusion, by Navarro and Baeza-Yates [2] and Consens and Milo [10].

To model three basic XML IR aspects, namely element relevance score computation, element score propagation, and element score combination, at the logical level of a database, we extended the original region algebra approaches with scoring operators and termed this new algebra Score Region Algebra (SRA). The overall goal of the score region algebra is to transparently model different aspects of the query formulation and search and answer element specification, and to support different retrieval models with different parameter specification applied to XML.

1.3 Outline

This paper is organized as follows. In the next section, we specify score region algebra used to define a framework for flexible and transparent XML ranked retrieval and illustrate how different retrieval models can be instantiated in score region algebra based on identified retrieval aspects. We present the experimental setup, including a description of our prototype system, and the evaluation results in Section 3. The paper is concluded with a short discussion and directions for future research.

2 Transparent Logical Algebra

In Section 2.1 we specify in more detail the three key aspects of XML IR. Next, we present four retrieval models that we use in Section 2.2. In Section 2.3 we define our score region algebra (SRA) for use at the logical level of databases to enable transparent specification of retrieval models.

2.1 Three aspects of XML IR

In XML IR query processing, three key aspects are identified. To explain this, we first discuss the three different parts in a typical NEXI query expression. Recall our example query from Section 1:

```
//article[about(.//abs, information retrieval)
   or about(.//abs, probabilistic database)]//sec[about(., language model)]
```

terms In our example query, we can identify six different query terms: 'information', 'retrieval', 'probabilistic', 'databases', 'language', and 'model'.

answer elements We can distinguish three different structural constraints[2] (i.e., element or tag name specifications): 'article', 'abs', and 'sec'. According to the NEXI specification, the answer element is the last element that has an *about* predicate specified, i.e., in our example the 'sec' element.

search elements All other elements which are not answer elements are called search elements. Within the search elements, we distinguish two different kinds: the lowest element inside an about, i.e., the last element in the *about* path expression, and other elements. There exists one special case where the lowest search element inside an about is '.', that refers to the search element that directly precede the about clause. Thus, it can happen that the search element is the answer element at the same time (e.g., the 'sec' element in our example query).

In our example query, the search elements are: 'article', 'abs', and 'sec'. The lowest search elements inside abouts are 'abs' and '.', i.e., 'sec'.

2.1.1 Element (relevance) score computation

The first task in XML ranked retrieval is to produce the relevance score for all nodes in the XML collection matching the lowest in about search elements. Following the NEXI specification, we consider all terms in isolation per lowest search element in the respective about clauses. In this section, we describe how to compute a score per search element-term pair. The combination of these scores is discussed in the next section.

In the example query, relevance scores have to be determined for the 'abs' elements with respect to the four terms: 'information', 'retrieval', 'probabilistic', and 'databases'. For the 'sec' element we have to compute scores for two terms: 'language' and 'model'. By employing a retrieval formula that specifies the relevance of a (part of) document given a query term (see, e.g., Equation 1 in Section 2.2), we can compute the 'abs' and 'sec' element relevance scores per term.

2.1.2 Element score combination

As about clauses typically can have more than one query term and scores are computed on a per term basis, those scores have to be combined on a per lowest search element basis. Depending on the preferred behavior, this can be seen as an OR or an AND combination. We can also distinguish cases where the NEXI query expresses an OR or an AND combination on arbitrary search elements.

In our example query, the first two *about* clauses contain two terms each. So we can interpret this either as 'abs' should be about 'information' AND 'retrieval' or 'abs' should be about 'information' OR 'retrieval'. It is up to the implementer of a specific model to make a choice, unless the user explicitly specified AND or OR in the NEXI expression (although a specific model might choose to ignore that).

In case of consecutive search elements in a NEXI path expression expressing an ancestor-descendant relationship, the ancestor search element can have multiple matching descendant search elements. Propagating scores between ancestors and descendant is called score propagation. This is discussed in the next section.

2.1.3 Element score propagation

Although it might seem unnecessary in our example as we have used it till now, the necessity of the propagation to the common ancestor element can be seen in the case of the following, different

[2]Here we assume a strict interpretation of structural constraints, although element names can be considered as a hint for the retrieval system. See, e.g., vague content-and-structure (VCAS) queries in INEX [17].

NEXI query:

```
//article[about(.//abs, information retrieval)
   and about(.//kwd, probabilistic database)]
```

To perform an AND-like combination of 'abs' and 'kwd' elements in this case, we need to propagate the scores to the common ancestor 'article' element.

We can define the element score propagation as the translation of scores to the ancestor or descendant elements where these scores can be combined based on the type of combination explicitly specified in NEXI. We can distinguish between two types of score propagation: upwards and downwards score propagation. For our original example query, the score should be propagated from 'abs' to 'article' elements that matches the upwards score propagation. This scenario happens if the NEXI predicate has logically combined multiple *abouts*, as in our NEXI query example:

```
//article[about(.//abs, information retrieval)
   or about(.//abs, probabilistic database)]
```

The second scenario is when the *about* clause contains at least one element selection. In the example with two element selections in the *about*:

```
//article[about(.//sec//p, information retrieval)]
```

scores need to be propagated from 'p' elements to 'sec' elements and then to 'article' elements.

In the case of downwards propagation scores should be propagated from search elements to the contained search or answer elements. In our example, the scores are propagated from the search elements 'article' to the 'sec' elements which are answer elements. If the section in not the answer element, i.e., if we have

```
//sec//p[about(., language model)]
```

instead of

```
//sec[about(., language model)]
```

in our query example, the scores should be further propagated downwards from 'sec' to 'p' elements.

2.2 Retrieval models

We have chosen four state of the art retrieval models to test the transparency of our approach. Here we briefly describe these retrieval models. The state of the art retrieval models are statistical language models [25], the Okapi (INQUERY) model [7, 36], the tf.idf model [38], and the Garden Point XML (GPX) model [18][3]. For language modeling approach we use three types: language models with smoothing, language models without smoothing, and language models with document weighting.

In these approaches, the relevance score of a document is based on the fact that documents that contain more occurrences of a term, i.e., have higher *term frequency*, are more important to the user. Additionally, to incorporate the significance of a term for ranked retrieval, these models also include background statistics. Background statistics are based on a number of terms in the whole collection, i.e., *collection frequency*, or number of documents containing a term, i.e., *document frequency*.

[3]Although this is not a well known retrieval model we have chosen it as it is among the most effective ones presented at INEX 2004 workshop: http://inex.is.informatik.uni-duisburg.de:2004/.

Language model In the language model approach (with smoothing), the relevance score of the document (doc) given the query terms (tm_i, $i = 1, 2, ..., n$, $tm_i \in q$) can be computed as:

$$S(doc|q) = \prod_{i=1}^{n} (\lambda \frac{tc(tm_i, doc)}{len(doc)} + (1 - \lambda) \frac{tc(tm_i, col)}{len(col)}) \quad (1)$$

where n is the number of terms (tm_i) in the query (q), $tc(tm_i, doc)$ and $tc(tm_i, col)$ denote the number of terms in the document doc or in the collection col respectively, $len(doc)$ and $len(col)$ are the lengths of the document and the collection respectively (i.e., the number of terms they contain), and λ is a smoothing parameter (ranging from 0 to 1) that specifies the relative influence of the foreground and background statistics in the final document ranking score computation. The language model approach without smoothing is just a special case of language modeling approach where $\lambda = 1$.

The language model with document weighting is the approach specific for retrieval tasks where retrieved entity (ent) is not equal to the document itself (such as XML elements), and therefore the information about term statistics per entity and term statistics per document containing the entity (doc) can be combined in the language modeling approach:

$$S(ent|q) = \prod_{i=1}^{n} (\alpha \frac{tc(tm_i, ent)}{len(ent)} + \beta \frac{tc(tm_i, doc)}{len(doc)} + (1 - \alpha - \beta) \frac{tc(tm_i, col)}{len(col)}) \quad (2)$$

Okapi The Okapi (INQUERY) retrieval model is based on the BM25 algorithm [36]:

$$S(doc|q) = \sum_{i=1}^{n} (ln \frac{N - dc(tm_i) + 0.5}{dc(tm_i) + 0.5} \cdot \frac{(k_1 + 1) \cdot tc(tm_i, doc)}{k_1((1 - b) + b \frac{len(doc)}{avdl}) + tc(tm_i, doc)} \cdot \frac{(k_3 + 1) \cdot tc(tm_i, q)}{k_3 + tc(tm_i, q)}) \quad (3)$$

where N is the total number of documents in the collection, $dc(tm_i)$ is the number of documents in the collection that contain term tm_i, $avdl$ is the average document length, $tc(tm_i, q)$ is the number of terms tm_i in the query q, and k_1 (between 1.0 and 2.0), k_3 (between 0 and 1000), and b (≈ 0.75) are constants.

Note that in the INQUERY system, several functions are implemented to combine the scores of single term relevance score computations, such as sum, multiplication, probabilistic interpretation (see [7]).

tf.idf For the tf.idf approach, we used the basic tf.idf formula specified in [38]:

$$S(doc|q) = \sum_{i=1}^{n} tc(tm_i, doc) \cdot ln \frac{N}{dc(tm_i, doc)} \quad (4)$$

The parameters of the formula are the same as in Equations 1 and 3.

GPX Finally, the basic formula in the GPX approach defines the relevance score of the document with respect to the query terms as[4]:

$$S(doc|q) = A^{n-1} \sum_{i=1}^{n} \frac{tc(tm_i, doc)}{tc(tm_i, col)} \quad (5)$$

where A is the parameter with a value between 3 and 10.

In the next section, after introducing score region algebra, we explain how these IR models can be applied to SRA.

[4]Note that the original formula defines score computation for leaf XML elements instead of documents [18].

2.3 Score region algebra

The application of the idea of text regions to XML documents is straightforward. Each XML document can be seen as a sequence of tokens, e.g., start tags, end tags, terms, etc., where each token can be indexed to model the XML tree structure (see, e.g. [32]), represented as a set of text regions. To be able to represent XML properly, the definition of a region in score region algebra is richer than in previous region algebra approaches. In the specification of our SRA data model we distinguish between different node types in XML documents in order to provide a uniform platform for defining the region algebra operators. Furthermore, we enriched the original model with region score attribute and introduced a number of operators for score manipulation. The logical data model of SRA is based on *region sets*, where each region is defined below.

Definition 1 *The SRA data model is defined on the domain R which represents a set of region tuples. Region tuple r ($r \in R$), $r = (s, e, n, t, p)$, is defined by these five attributes: region start attribute - s, region end attribute - e , region name attribute - n, region type attribute - t, and region score attribute - p. Region start and end attributes must satisfy ordering constraints ($e_i \geq s_i$). If $\prec$ denotes the equivalence $r_i \prec r_j \Leftrightarrow s_j < s_i \leq e_i < e_j$, we can state that for two arbitrary regions in SRA it is either $r_i \prec r_j$, $r_i \equiv r_j$, $r_j \prec r_i$, or they are not comparable, i.e., one region preceeds the other. Furthermore, each region in the SRA data model is unique.*

The semantics of region start and region end attributes are the same as in other region algebra approaches: they denote the bounds of a region. The region name attributes are used to denote node names, content words, attribute names, attribute values, etc. To distinguish between different name "roles" in XML we used the region type attribute. We used *node* for the element node in XML, *text* for the text node, *term* for the term present in a text node, etc. Finally, the region score information item is used to specify the relevance score of a region with respect to a given query.

Table 1: Score region algebra operators.

Operator	**Operator definition**
$\sigma_{n=name,t=type}(R)$	$\{r \mid r \in R \wedge n = name \wedge t = type\}$
$\sigma_{\diamond num}(R_1)$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in C \wedge t_2 = term \wedge r_2 \prec r_1 \wedge n_2 \diamond num\}$, where $\diamond \in \{=, <, >, \leq, \geq\}$
$R_1 \sqsupset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}$
$R_1 \sqsubset R_2$	$\{r_1 \mid r_1 \in R_1 \wedge \exists r_2 \in R_2 \wedge r_1 \prec r_2\}$
$R_1 \sqsupset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge t_1 = node \wedge t_2 = term \wedge p := f_{\sqsupset}(r_1, R_2)\}$
$R_1 \not\sqsupset_p R_2$	$\{r \mid r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge t_1 = node \wedge t_2 = term \wedge p := f_{\not\sqsupset}(r_1, R_2)\}$
$R_1 \blacktriangleright R_2$	$\{r \mid r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleright}(r_1, R_2)\}$
$R_1 \blacktriangleleft R_2$	$\{r \mid r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge t_1 = node \wedge t_2 = node \wedge p := f_{\blacktriangleleft}(r_1, R_2)\}$
$R_1 \sqcap_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1, e_1, n_1, t_1) = (s_2, e_2, n_2, t_2) \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := p_1 \otimes p_2\}$
$R_1 \sqcup_p R_2$	$\{r \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s, e, n, t) := (s_1, e_1, n_1, t_1) \vee (s, e, n, t) := (s_2, e_2, n_2, t_2)) \wedge p := p_1 \oplus p_2\}$

The aim of SRA is to support ranked retrieval as a part of the algebra, and not as a side-effect. This distinguishes it from other region algebra proposals that include ranked retrieval (e.g., [5]). The basic SRA operators are defined in Table 1. In the specification of region algebra operators we use R_i $(i = 1, 2, ...)$ to denote the region sets, their corresponding non-capitals to denote regions in these region sets (r_i), and corresponding indexed non-capitals to denote region attributes $(s_i, e_i, n_i, t_i, p_i)$. With C we denote the set of all regions in the collection and with *Root* we denote the (artificial) root element for the whole collection.

The operators in SRA take one or two region sets as operands and produce a region set as result. The first four operators enable Boolean selection of regions based on their attributes or containment

relations. The selection operator, σ, has two variants. The first one ($\sigma_{n=name,t=type}(R)$) specifies the selection based on name and type attributes[5]. The second selection operator selects regions that contain a term region in which content (casted to a number) is equal, greater or equal, less or equal, greater or less than the number specified (num). The last two selection operators select regions based on their containment relations, i.e., regions that contain other regions ($\sqsupset$), or regions that are contained in other regions ($\sqsubset$).

The other six operators specify score manipulation among regions. To enable the instantiation of different retrieval models, they are defined using four abstract scoring functions: $f_{\sqsupset}$, $f_{\not\sqsupset}$, $f_{\blacktriangleright}$, and $f_{\blacktriangleleft}$, and two abstract operators: $\otimes$ and $\oplus$. These abstract functions and abstract operators model three aspects of XML IR. They are specified based on auxiliary functions that count the number of regions in the region set R, denoted with $|R|$, compute the size of the region r:

$$size(r) = e - s - 1 \tag{6}$$

and compute the average size of the regions with the region name n in the collection, denoted with $avg\_size(r)$.

In the following we explain how the three XML retrieval aspects are modeled using abstract functions and operators.

2.3.1 Element (relevance) score computation

Operators $\sqsupset_p$ and $\not\sqsupset_p$ model element relevance score computation, where $\sqsupset_p$ models the concept that the search elements (regions in the first operand) should contain the term (region) and $\not\sqsupset_p$ models the concept that the search elements should not contain the term. Therefore, the functions $f_{\sqsupset}(r_1, R_2)$ and $f_{\not\sqsupset}(r_1, R_2)$, applied to a region r_1 and region set R_2, should result in the numeric value that specifies the relevance of the region (element) r_1 given the term regions in R_2 that it contains. Following the specification of four models in the previous section, we have different specifications of these abstract functions.

The language model (LM) can be instantiated in functions $f_{\sqsupset}(r_1, R_2)$ and $f_{\not\sqsupset}(r_1, R_2)$ based on Equation 1 and auxiliary functions as:

$$f_{\sqsupset}^{LM}(r_1, R_2) = p_1 \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1 - \lambda) \frac{|R_2|}{size(Root)}) \tag{7}$$

$$f_{\not\sqsupset}^{LM}(r_1, R_2) = p_1 \cdot (1 - (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1 - \lambda) \frac{|R_2|}{size(Root)})) \tag{8}$$

For a language model without smoothing, the relevance score computation function is specified using the same equations where $\lambda = 1$. Furthermore, for the third variant of the language model that uses document term statistics, we assume that there is a predefined XML element that correspond to the notion of a document in traditional IR, and we denote it with 'doc'. The language model with document weighting can be instantiated as:

$$f_{\sqsupset}^{LMA}(r_1, R_2) = p_1 \cdot (\alpha \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + \beta \frac{\sum_{r_2 \in R_2 | r \in C \wedge n = `doc' \wedge r_2 \prec r} p_2}{size(r)} + (1 - \alpha - \beta) \frac{|R_2|}{size(Root)}) \tag{9}$$

[5]Leaving the selection criterion for one of the attributes unspecified corresponds to a wild-card, i.e., $\sigma_{t=node}$ will select all regions that have an XML element node type, regardless of their name attribute.

$$f_{\not\sqsupset}^{LMA}(r_1, R_2) = p_1 \cdot (1 - (\alpha \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + \beta \frac{\sum_{r_2 \in R_2 | r \in C \wedge n = `doc' \wedge r_2 \prec r} p_2}{size(r)} + (1 - \alpha - \beta) \frac{|R_2|}{size(Root)})) \quad (10)$$

Similarly, we can instantiate these functions for the other retrieval models. In the following we give only the specification of function $f_{\sqsupset}$ since function $f_{\not\sqsupset}$ can be easily expressed as can be seen in Equations 7 to 10. Therefore, if $f_{\sqsupset}(r_1, R_2) = p_1 \cdot g(s_1, e_1, n_1, t_1, R_2)$, then $f_{\not\sqsupset}(r_1, R_2) = p_1 \cdot (1 - g(s_1, e_1, n_1, t_1, R_2))$, where $g(s_1, e_1, n_1, t_1, R_2)$ need to be normalized to a value between 0 and 1.

In the Okapi system we simplified the Equation 3 by removing the third fraction in the sum as it is based on a size of the query and is not supported in other models. The complex function $f_{\sqsupset}$ is specified as:

$$f_{\sqsupset}^{\text{Okapi}}(r_1, R_2) = p_1 \cdot ln \frac{|\{r \in C | n = n_1\}| - |\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}| + 0.5} \cdot \frac{(k_1 + 1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{k_1((1 - b) + b \frac{size(r_1)}{avg\_size(r_1)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2} \quad (11)$$

For the tf.idf approach we have[6]:

$$f_{\sqsupset}^{\text{tf.idf}}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \cdot ln \frac{|\{r \in C | n = n_1\}|}{|\{r \in C | n = n_1 \wedge \exists r_2 \in R_2 \wedge r_2 \prec r_1\}|} \quad (12)$$

The element relevance score computation in GPX model is specified based on Equation 5 as:

$$f_{\sqsupset}^{\text{GPX}}(r_1, R_2) = p_1 \; op \; \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{|R_2|} \quad (13)$$

In our experiments we use two variants of the GPX model, one where *op* is implemented as '+' with the default element region score 0, referred to as basic model (denoted with '*' in Table 2), and the other where *op* is implemented as '$\cdot$' with the default element region score 1 (see Section 3.2).

2.3.2 Element score propagation

The operators $\blacktriangleright$ and $\blacktriangleleft$ specify propagation of scores to the containing or contained elements, respectively. Thus, functions $f_{\blacktriangleright}(r_1, R_2)$ and $f_{\blacktriangleleft}(r_1, R_2)$ specify whether the propagation is performed with or without normalization, are the score values of contained or containing elements summed, averaged or maximized, etc.

For the language modeling approach with and without smoothing, as well as for tf.idf and Okapi model, we decided to use the the same approach for the basic experimental runs. We employed a weighted sum normalized by the size of regions in the first operand for modeling upwards element score propagation. The simple sum of scores is used for downwards element score propagation. We made this choice because these functions showed good results in our experiments [33].

[6]Note that the tf.idf approach can be considered as a simplified version of the approach taken by Mass and Mandelbrod [30], avoiding document pivot factor and complex score combinations.

$$f_{\blacktriangleright}^{LM,LMA,tf.idf,Okapi}(r_1, R_2) = p_1 \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \cdot size(r_2)}{size(r_1)} \tag{14}$$

$$f_{\blacktriangleleft}^{LM,LMA,tf.idf,Okapi}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{15}$$

However, for basic GPX model (denoted with '*' in Table 2) we employ different computations for element score propagation:

$$f_{\blacktriangleright}^{\text{GPX}}(r_1, R_2) = p_1 + \sum_{r_2 \in R_2 | r_1 \prec r_2} p_2 \tag{16}$$

$$f_{\blacktriangleleft}^{\text{GPX}}(r_1, R_2) = p_1 + \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{17}$$

We also tried Equation 14 and Equations 15 for additional GPX runs with default element region score 1, as well as sum instead of weighted sum for upwards score propagation in other models, defined similar to Equation 15 (see Section 3.2).

2.3.3 Element score combination

The abstract operator $\otimes$ specifies how scores are combined in an AND expression, denoted in SRA by $\sqcap_p$, while the operator $\oplus$ defines score combination in an OR expression, denoted in SRA with $\sqcup_p$. In our retrieval models for basic experimental series, we decided to make different choices for each model. For language models we used the instantiation where $\otimes$ is implemented as a product and $\oplus$ is implemented as a sum. For the tf.idf model, $\otimes$ is modeled as min and $\oplus$ is modeled as max. Following [7], in the Okapi (INQUERY) model we defined these two abstract operators as follows:

$$p_1 \otimes p_2 = p_1 \cdot p_2 \tag{18}$$

$$p_1 \oplus p_2 = 1 - (1 - p_1) \cdot (1 - p_2) \tag{19}$$

Due to a somewhat different specification of the GPX model with respect to other models (see Equations 5 and 13), we instantiated $\otimes$ as well as $\oplus$ as:

$$p_1 \otimes p_2 = p_1 \oplus p_2 = \begin{cases} p_1 + p_2 & \text{if } p_1 = 0 \vee p_2 = 0 \\ A \cdot (p_1 + p_2) & \text{otherwise} \end{cases} \tag{20}$$

Although this formula results in a retrieval model that is slightly different than the one given in Equation 5, it follows the semantics of the model which is to boost the scores for regions that contain more query terms.

Furthermore, we experimented with different implementations of score combination functions for each of the retrieval model instantiations as can be seen in Section 3.1.3.

3 Experiments

In this section, we describe our prototype system, the XML document collection, and the query set we used to demonstrate the functionality or our approach (Section 3.1). Next, in Section 3.2, we show and discuss the results of our experimental runs.

3.1 Setup

To demonstrate the usefulness of our approach, we have built a prototype system and evaluated it against a well-known document collection and query set. Below, we first describe out prototype and then continue with some details about the data and query set.

3.1.1 Prototype system

Following good practice in database systems design, we setup our prototype following a typical three-layered architecture:

Conceptual layer This layer takes a NEXI query expression as input, puts it through a filter to standardize/sanitize[7] it, and produces an SRA expression to be fed into the next layer.

Logical layer This layer takes an SRA expression as input. Like the previous layer, it does some filtering and standardization and transforms it into an expression for input to the next layer.

Physical layer For the physical implementation we use a low-level physical DB engine - MonetDB [4][8]. The score region algebra operators are implemented as a set of procedures in Monet Interpreter Language (MIL).

Due to a modular setup, changing retrieval models is straightforward: one just plugs in a different transformation module and implements the supporting retrieval functions in MIL. Building a new module, to capture yet another retrieval model to experiment with, takes only half an hour.

The hardware platform used[9] to produce the results presented in this section is an old PC running Linux 2.4.20-8smp, with two Pentium™ III 600 MHz CPUs, 1 GB of main-memory, and a 100 GB disk array mounted in RAID 0 (striping) mode.

3.1.2 Document collection, query set, and evaluation

Not many XML document collections exist that also come with IR queries and user assessments to evaluate the retrieval effectiveness of a system. We used the most well known collection, provided by the INEX initiative. This collection consists of IEEE journal papers (articles) in XML format.

Each year a new set of so-called topics is constructed by the participants of the INEX workshop series. Those topics contain a NEXI query expression and a textual description of the so-called information need of the user, i.e., an explanation of what kind of answers should be considered as good results for that query. Every participant runs the queries on their system.

By pooling and manual reviewing by the participants the results are assessed, i.e., checked whether they are good answers or not. These assessments are then aggregated by the organization and made available to the participants. Using a tool provided by the organization each participant than can compute how good their system is performing in terms of precision and recall.

In our experiments we wanted to test our system against the queries (topics) that include search terms as well as structural constraints, termed content-and-structure (CAS) topics in INEX. We used the 30 CAS topics and corresponding assessments of 2003 and 36 CAS topics and corresponding assessments of 2004 (see [39] and Appendix in [17]) to test our architecture for each of the models described in Section 2. In Section 3.2 we present the results of these experiments.

[7] In our case the standardization consists of removing the '"' characters denoting phrases, '+' modifiers query terms and terms with '-' modifiers denoting important and unimportant terms. Note that our system can handle phrases and term modifiers [33] though we left it out as it goes beyond the scope of this paper.

[8] At the same time we developed the PostreSQL implementation but we used the MonetDB implementation for our experiments since it was faster and less resource demanding.

[9] Although wall clock performance is not really the issue of this paper, we mention this for completeness. See Appendix B for more details on performance.

3.1.3 Experiment series

We performed several experiments using the settings described above. First of all we ran the system for the various score computation models described in Section 2.3. For some of these models we also ran experiments with varying score propagation and combination functions as described in Sections 2.3.2 and 2.3.3.

In our runs we used standard values for the various parameters in different models, except for the language model with document weighting. In our experiments for language model without smoothing $\lambda = 0.5$, for Okapi $k_1 = 1.5$ and $b = 0.75$, and for GPX $A = 5$. For language model with article weighting we considered that documents correspond to 'article' elements (i.e., 'doc' is actually 'article'), and we performed experiments with varying values of α and β: $\alpha = 0.1$ and $\beta = 0.5$, $\alpha = 0.1$ and $\beta = 0.3$, and $\alpha = 0.2$ and $\beta = 0.6$.

We run the experiments on INEX 2003 topics and to test the consistency of results we repeat the experiments on 2004 topics. Also, for 2003 queries we run the experiments that explore the usage of '+' and '-' modifiers (for details on the implementation see [33]) but as they did not differ significantly from the original 2003 runs we are not going to discuss them here and we give only the resulting recall precision graphs in Appendix C.3.

3.2 Results

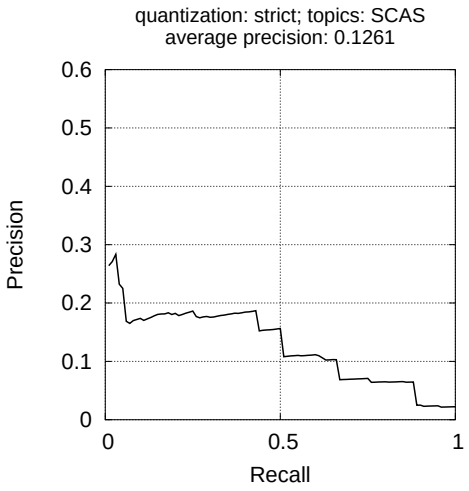
In this section we discuss the results of the experiments described above using the topics from INEX 2003 and 2004. In Appendix A we listed the NEXI expressions of all topics from INEX 2003 and 2004[10].

Table 2: Experimental series for INEX 2003 topics.

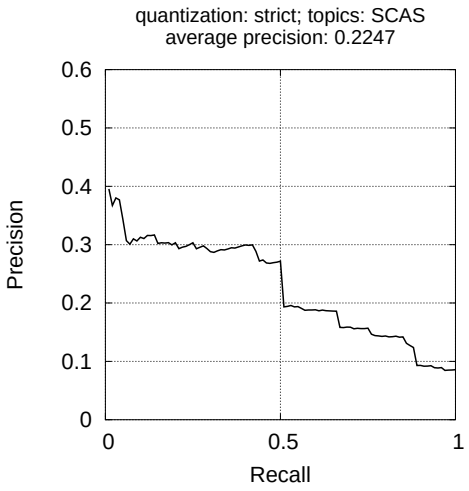
Series	Score function	($f_\blacktriangleright$) Propagation	AND ($\otimes$)	OR ($\oplus$)	MAP
I	LM, $\lambda = 1$	weighted sum	product	sum	**0.1261**
I^a	LM, $\lambda = 1$	weighted sum	sum	sum	0.1216
II	LM, $\lambda = 0.5$	weighted sum	product	sum	0.2247
II^a	LM, $\lambda = 0.5$	sum	product	sum	**0.2367**
II^b	LM, $\lambda = 0.5$	weighted sum	sum	sum	0.1201
III	LMA, $\alpha = 0.1$, $\beta = 0.5$	sum	product	sum	**0.2497**
III^a	LMA, $\alpha = 0.1$, $\beta = 0.3$	sum	product	sum	0.2488
III^b	LMA, $\alpha = 0.2$, $\beta = 0.6$	sum	product	sum	0.2488
IV	Okapi, $k_1 = 1.5$, $b = 0.75$	weighted sum	product	prob. sum	0.1351
IV^a	Okapi, $k_1 = 1.5$, $b = 0.75$	sum	sum	sum	0.2358
IV^b	Okapi, $k_1 = 1.5$, $b = 0.75$	weighted sum	sum	sum	**0.2578**
V	tf.idf	weighted sum	min	max	0.1425
V^a	tf.idf	weighted sum	sum	sum	0.1509
V^b	tf.idf	sum	sum	sum	0.1561
V^c	tf.idf	sum	product	prob.sum	**0.1594**
VI	GPX*, $A = 5$	sum*	exp. sum	exp. sum	**0.2782**
VI^a	GPX, $A = 5$	sum	exp. sum	exp. sum	0.2778
VI^b	GPX, $A = 5$	weighted sum	exp. sum	exp. sum	0.2519

In Figure 1 we show the recall precision graphs for our basic retrieval models (experimental series), aggregated over all 30 INEX 2003 topics. The complete overview of recall precision graphs for performed runs is placed in Appendix C.1. The mean average precision (MAP) for basic (I to VI) and additional experimental series (denoted with ^{a}, ^{b}, and ^{c}) is given in the last column of

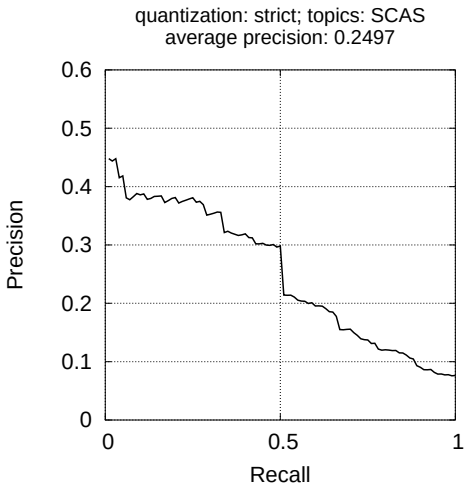
[10]Note that the numbering starts at 61 and 127 as each year the new topics for INEX are added to the already existing list, including the topics that do not have structural constraints, termed content-only (CO) topics in INEX, e.g., in 2004 topics 91 to 126.



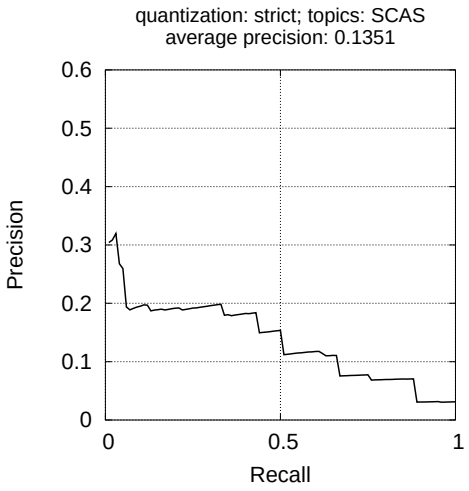
(a) Language model without smoothing



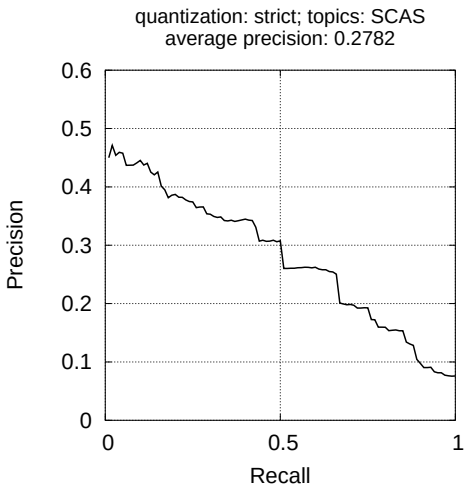
(b) Language model with smoothing



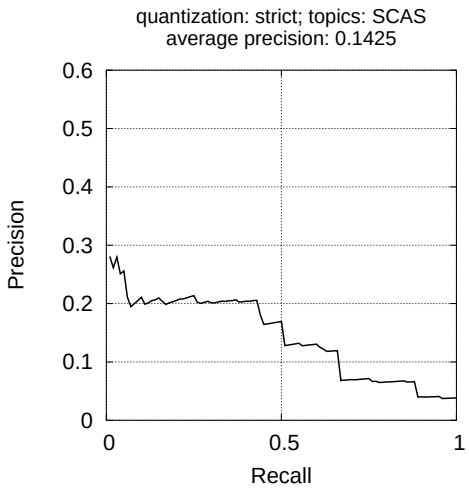
(c) language model with document weighting



(d) Okapi model

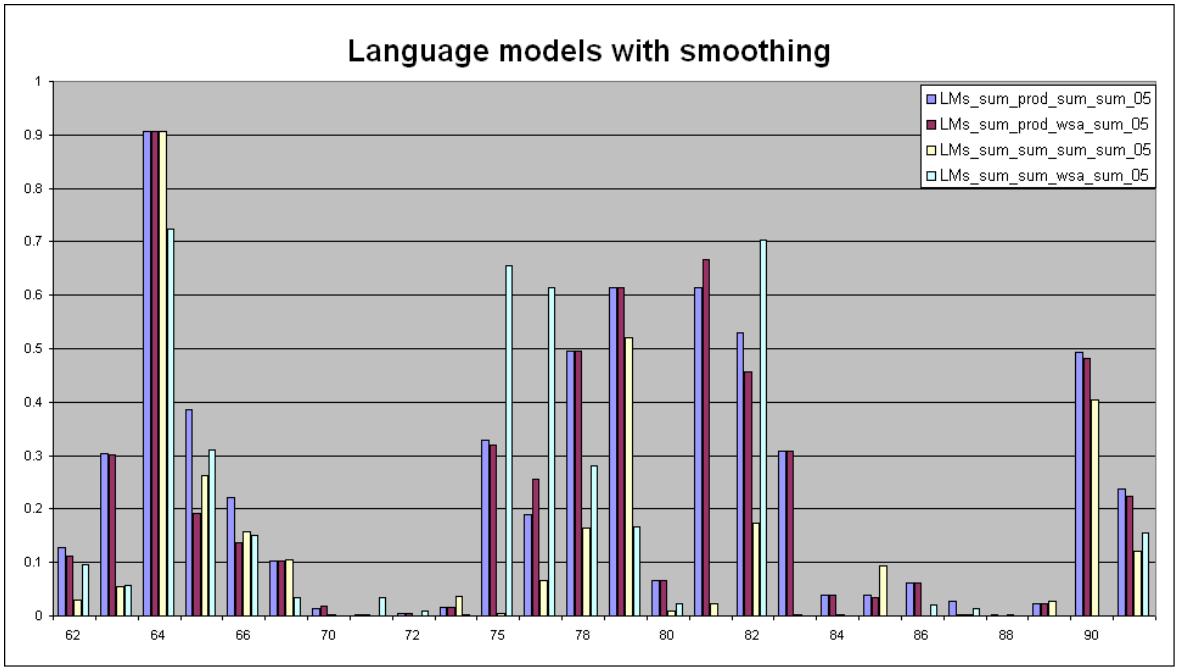


(e) GPX model

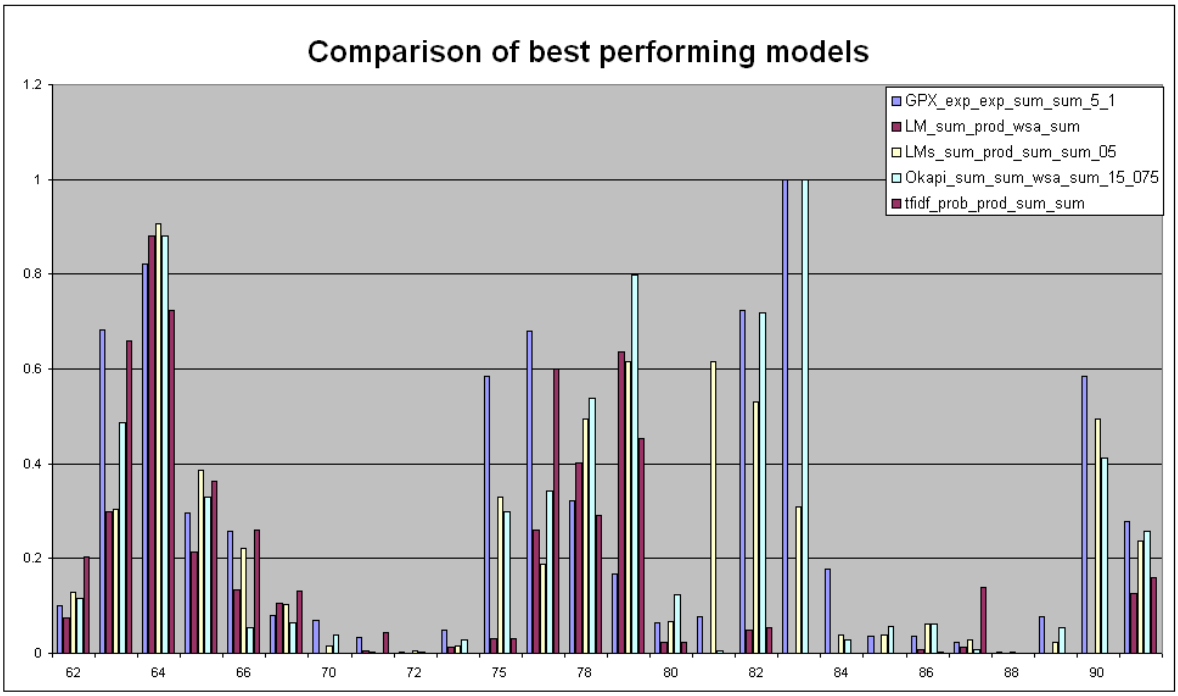


(f) tf.idf model

Figure 1: Recall precision graphs for basic retrieval models on INEX 2003 topics.



a)



b)

Figure 2: Comparison of mean average precision (MAP) values for **a)** variants of language models, and **b)** best performing models on INEX 2003 topic set.

Table 2. The mean average precision is actually the average precision aggregated over all topics. To produce recall precision graphs and compute mean average precision we use the official INEX tool for the evaluation (see [17] for details). The highest mean average precision for each score computation model is given in bold.

As can be seen in Table 2[11] the 2003 results depend a lot on the function used for relevance score computation. The best runs for language models with smoothing, Okapi, and GPX significantly outperform language models without smoothing and tf.idf. However, for language models with smoothing, Okapi, and GPX, mean average precision is quite different for different combinations of score propagation and combination functions. For example, the mean average precision decreases for almost 100% if we compare series II (0.2247) and II^b (0.1201) for language models and increases for more than 100% in series IV (0.1351) and IV^b (0.2578) for Okapi. On the other hand, no matter what kind of functions for score propagation and score combination we use the mean average precision for tf.idf models is approximately 0.15 (see series V, V^a, V^b, and V^c).

If we look at the mean average precision per topic we can see that even for the same score computation model the results are significantly different if we use different score combination and propagation functions. This can be seen in Figure 2a where the MAP for different variants of language models with smoothing are displayed. Furthermore, in Figure 2b we can see that some models perform better for some topics and worse for others. For example, the overall worst performing tf.idf model outperforms the best GPX model for topics 68 and 87.

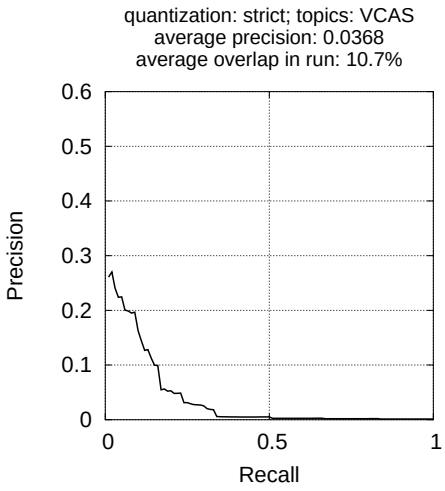
To test the consistency of the results obtained using INEX 2003 test collection we repeated the same experiments on INEX 2004 test collection. However, the assessments on INEX 2004 topics are done differently than on 2003. This is due to the vague interpretation of structural constraints for INEX 2004 topics, i.e., the usage of vague content-and structure (VCAS) queries (see Appendix in [17]). In our experiments we interpreted these constraints as a strict constrains and therefore significantly reduced the recall base for our result evaluation. As a consequence, our system was not able to find all the relevant answers and in most of the precision-recall graphs (see Figure 3) the precision drops almost at 0 at recall levels around 0.5, and the mean average precision is significantly lower than for 2004 queries as can be seen in the last column in Table 3.

Table 3: Experimental series for INEX 2004 topics.

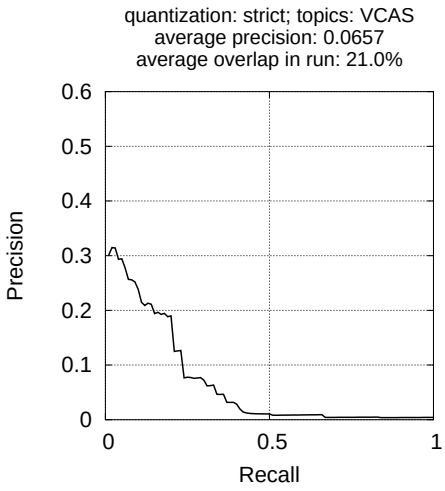
Series	**Score function**	**($f_\blacktriangleright$) Propagation**	**AND ($\otimes$)**	**OR ($\oplus$)**	**MAP**
I	LM, $\lambda = 1$	weighted sum	product	sum	0.0290
I^a	LM, $\lambda = 1$	weighted sum	sum	sum	**0.0368**
II	LM, $\lambda = 0.5$	weighted sum	product	sum	0.0657
II^a	LM, $\lambda = 0.5$	sum	product	sum	**0.0694**
II^b	LM, $\lambda = 0.5$	weighted sum	sum	sum	0.0398
III	LMA, $\alpha = 0.1$, $\beta = 0.5$	sum	product	sum	0.0563
III^a	LMA, $\alpha = 0.1$, $\beta = 0.3$	sum	product	sum	0.0577
III^b	LMA, $\alpha = 0.2$, $\beta = 0.6$	sum	product	sum	**0.0593**
IV	Okapi, $k_1 = 1.5$, $b = 0.75$	weighted sum	product	prob. sum	0.0192
IV^a	Okapi, $k_1 = 1.5$, $b = 0.75$	sum	sum	sum	0.0462
IV^b	Okapi, $k_1 = 1.5$, $b = 0.75$	weighted sum	sum	sum	**0.0513**
V	tf.idf	weighted sum	min	max	0.0286
V^a	tf.idf	weighted sum	sum	sum	**0.0520**
V^b	tf.idf	sum	sum	sum	0.0419
V^c	tf.idf	sum	product	prob.sum	0.0302
VI	GPX*, $A = 5$	sum*	exp. sum	exp. sum	0.0587
VI^a	GPX, $A = 5$	sum	exp. sum	exp. sum	0.0638
VI^b	GPX, $A = 5$	weighted sum	exp. sum	exp. sum	**0.0653**

From Table 3 we can conclude that all three subtypes of language models and Okapi based

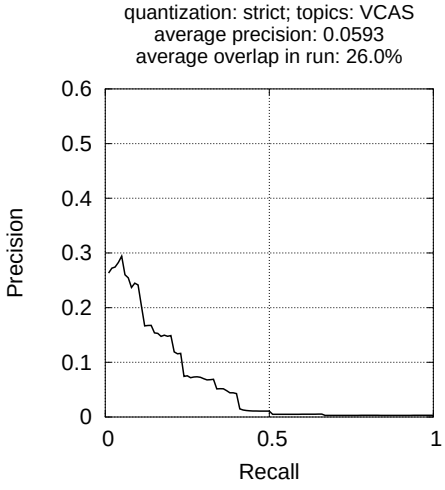
[11] The '*' denotes the GPX run where the default region score is 0.



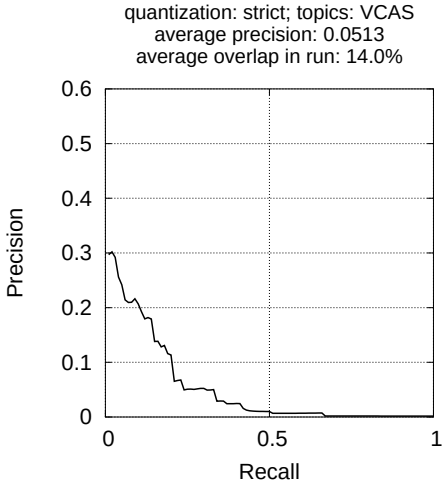
(a) Language model without smoothing



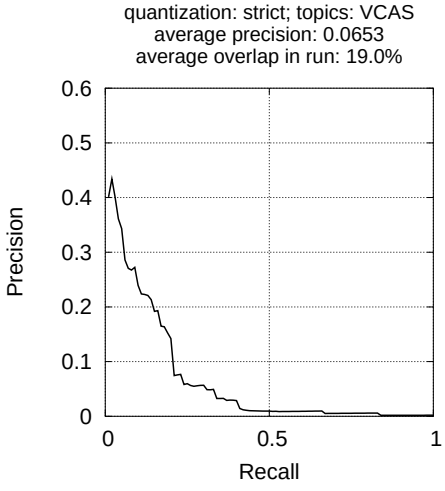
(b) Language model with smoothing



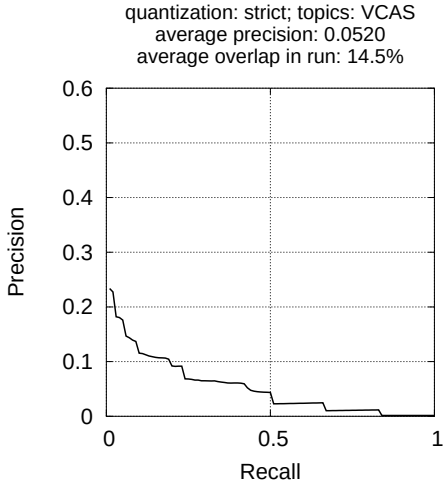
(c) language model with document weighting



(d) Okapi model



(e) GPX model



(f) tf.idf model

Figure 3: Recall precision graphs for best performing retrieval models on INEX 2004 topics.

models show consistent behavior on 2003 and 2004 topics, although the LMA model with parameters $\alpha = 0.2$ and $\beta = 0.6$ slightly outperforms two other parameter combinations. However, for tf.idf and GPX runs the mean average precision differs from 2003 runs. This is especially the case for tf.idf since the V^a run (0.0520) significantly outperforms the other three runs (0.0286, 0.0419, and 0.0302). The recall precision graphs for the best 2004 runs can be seen in Figure 3, and the complete set of recall precision graphs is presented in Appendix C.2.

Taking into account different experimental series we can see that the language model with smoothing (0.0694) outperforms GPX (0.0653) and Okapi (0.0513) on 2004 topics. Also, the tf.idf run V^a has relatively high mean average precision (0.0520), even higher than the Okapi run IV^b, whereas for the 2003 topics the same Okapi run was far better (0.2578 against 1509). This could be the consequence of different assessments as well as different query types that used in INEX 2004. With the results of these experiments we have shown that we need to study in more detail the differences of these models and to analyze test collection, i.e., to classify topics and clarify the assessments process, if we want to get clearer picture of XML IR.

Furthermore, in our experiments the best results are obtained by using the GPX model for 2003 topics (0.2782) and language models for 2004 topics (0.0694). However, the question is whether some of the other models models for score computation with the right choice for the score combination and score propagation functions and the right value for parameters, λ, α, β, k_1, b, and A, can boost the mean average precision and outperform GPX for 2003 topics and language models for 2004 topics. This points out that further study need to be taken to understand better the term distribution among XML elements and for specifying the best score computation, score combination and score propagation functions.

Additionally, the effectiveness of each model can be studied with respect to each topic in isolation, to determine which retrieval model is the most appropriate for each topic. This could help us to classify topics based on their features, such as number of query terms, existence of upwards or downwards score propagation, type of the query with respect to the expected answer, etc., and apply the best retrieval model for each topic type. We hope to answer these questions in our future research.

Finally, in this paper we did not address the problem of element overlap in the result set. However, in Figure 3 and in Appendix C.2 we report the overlap computed as specified in [11]. The overlap was recently identified as a potential problem for evaluation of retrieval systems [28], and despite several proposals for new metrics that take into account the overlap properties [12, 26, 28] not many solutions were suggested for efficiently removing overlap in result set without affecting the system's effectiveness. Therefore, the overlap problem is still an open question for XML IR that demands further attention.

4 Conclusions and future work

Most XML-IR systems adapt and extend existing flat file IR systems to support the searching of structured XML documents. Since these approaches are retrieval model specific and depend on the physical implementation, it is difficult to adapt them to support different retrieval models. We believe that existing XML-IR database approaches have to be made *transparent* in order to satisfy complex user information needs expressed on top of XML collections. We argue that the right architectural level to achieve this transparency is the logical level. In that way the system is also flexible for different aspects of IR search over XML at the conceptual level and distinct implementations of storage schemes and access algorithms at the physical level. By developing a transparent score region algebra at the logical level of a flexible three-level database system we are able to support the application of state of the art IR models to ranked XML retrieval. Also, it provides us with a uniform framework where we can compare the effectiveness and study properties of different XML retrieval models as we have shown in this paper.

We are planning to further investigate the usefulness of the transparent logical algebra by

applying different models to XML IR and to study the properties of score region algebra operators with respect to their consistency in ranking and their efficiency. We are also concerned with the modeling of term modifiers ('+' and '−'), explicit term and element weights, and phrases in score region algebra (see, e.g., [33]). Furthermore, we aim to better understand stemming and synonyms for terms, and vague treatment of search and answer elements throughout the instantiation of retrieval models in the SRA. Finally, we aim to investigate which combination of scoring functions in score region algebra operators is more appropriate for different topic types, such as topics with same search and answer elements, queries without upward or downward propagation, etc. This will guide us to a more effective and efficient, transparent XML-IR database system.

References

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th conference on World Wide Web*, pages 583–594, 2004.

[2] R. Baeza-Yates and G. Navarro. Proximal Nodes: A Model to Query Document Databases by Content and Structure. In *ACM Transactions on Information Systems 15 (4)*, volume 15, pages 401–435, 1997.

[3] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML Query Language. Technical report, W3C, 2002.

[4] P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications*. PhD thesis, CWI, 2002.

[5] F.J. Burkowski. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Texts. In *Proceedings of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–125, 1992.

[6] S. Buxton and M. Rys. XQuery and XPath Full-Text Requirements. Technical report, W3C, 2003.

[7] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of 3rd International Conference on Database nad Expert Systems (DEXA)*, pages 78–83, 1992.

[8] J. Clark and S. DeRose. XML Path Language XPath Version 1.0. Technical report, W3C, 1999.

[9] C.L.A. Clarke, G.V. Cormack, and F.J. Burkowski. An Algebra for Structured Text Search and a Framework for its Implementation. *The Computer Journal*, 38(1):43–56, 1995.

[10] M. Consens and T. Milo. Algebras for Querying Text Regions. In *Proceedings of the ACM Conference on Principles of Distributed Systems*, pages 11–22, 1995.

[11] A.P. de Vries, G. Kazai, , and M. Lalmas. Evaluation Metrics 2004. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[12] A.P. de Vries, G. Kazai, and M. Lalmas. Tolerance to Irrelevance: A User-effort Oriented Evaluation of Retrieval Systems without Predefined Retrieval Unit. In *RIAO Conference Proceedings*, pages 463–473, 2004.

[13] D. Florescu and I. Manolescu. Integrating Keyword Search into XML Query Processing. In *Proceedings of the 9th International World Wide Web Conference*, pages 67–76, 2000.

[14] N. Fuhr. Models for Integrated Information Retrieval and Database Systems. *IEEE data engineering bulletin*, 19(1):3–13, 1996.

[15] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, 2001.

[16] N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM TOIS*, 22(2):313–356, 2004.

[17] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, ERCIM Publications, 2004.

[18] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX 2003), to appear*, ERCIM Publications, 2004.

[19] N. Gövert, M. Abolhassani, N. Fuhr, and K. Großjohan. Content-oriented XML Retrieval with HyRex. In *Proceedings of the First Workshop of the INitiative for the Evaluation of XML retrieval (INEX 2002)*, ERCIM Publications, 2003.

[20] T. Grabs and H.-J. Shek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In *Proceedings of the XML and Information Retrieval Workshop 25th ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[21] D.A. Grossman and O. Frieder. *Information retrieval: algorithms and heuristics*. The Kluwer international series in engineering and computer science. Kluwer Academic, Boston, 1998. ISBN 0-7923-8271-4.

[22] T. Grust. Accelerating XPath Location Steps. In *Proceedings of the 21st ACM SIGMOD International Conference on Management of Data*, pages 109–120, 2002.

[23] T. Grust, S. Sakr, and J. Teubner. XQuery on SQL Hosts. In *Proceedings of the 30th Int'l Conference on Very Large Data Bases (VLDB)*, 2004.

[24] L. Guo, S. Feng, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proceedings of ACM SIGMOD*, 2003.

[25] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Twente, The Netherlands, 2001.

[26] K. Järvelin and J. Kakäläinen. Cumulated Gain-based Evaluation of IR Techiques. In *ACM Transactions on Information Systems*, volume 20(4), pages 551–556, 2002.

[27] J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length normalization in XML retrieval. In *Proceedings of the 27th ACM SIGIR conference on research and development in information retrieval (SIGIR 2004)*, pages 80–87, 2004.

[28] G. Kazai, M. Lalmas, , and A.P. de Vries. The Overlap Problem in Content-oriented XML Retrieval Evaluation. In *Proceedings of the 27th ACM SIGIR Conference*, pages 72–79, 2004.

[29] J. List, V. Mihajlović, A. de Vries, G. Ramirez, and D. Hiemstra. The TIJAH XML-IR System at INEX 2003. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX 2003)*, ERCIM Workshop Proceedings, 2004.

[30] Y. Mass and M. Mandelbrod. Component Ranking and Automatic Query Refinement for XML Retrieval. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX), to appear*, 2005.

[31] V. Mihajlović, H.E. Blok, D. Hiemstra, and P.M.G. Apers. Score Region Algebra: Building a Transparend XML-IR Database. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2005.

[32] V. Mihajlović, D. Hiemstra, H. E. Blok, and P. M. G. Apers. An XML-IR-DB Sandwich: Is it Better with an Algebra in Between? In *Proceedings of the SIGIR workshop on Information Retrieval and Databases (WIRD'04)*, pages 39–46, 2004.

[33] V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX), to appear*, 2005.

[34] P. Ogilvie and J. Callan. Using Language Models for Flat Text Queries in XML Retrieval. In *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, ERCIM Publications, 2004.

[35] J. Pehcevski, J. A. Thom, and A-M. Vercoustre. RMIT INEX Experiments: XML Retrieval Using Lucy/eXist. In *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX 2003)*, ERCIM Publications, 2004.

[36] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of 17th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

[37] A. Salminen and F.W. Tompa. PAT Expressions: An Algebra for Text Search. In *Proceedings of the 2nd International Conference in Computational Lexicography, COMPLEX'92*, pages 309–332, 1992.

[38] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGrow-Hill, New York, NY, USA, 1st edition, 1983.

[39] A. Trotman and R. A. O'Keefe. The Simplest Query Language That Could Possibly Work. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the Second Workshop of the INitiative for the Evaluation of XML retrieval (INEX 2003)*, ERCIM Publications, 2004.

[40] D. Tsichritzis and A. Klug. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information systems*, 3:173–191, 1978.

[41] S. R. Vasanthakumar, J. P. Callan, and W. Bruce Croft. Integrating INQUERY with an RDBMS to Support Text Retrieval. *IEEE data engineering bulletin*, 19(1):24–33, 1996.

[42] A.P. de Vries, M.G.L.M. van Doorn, H.M. Blanken, and P.M.G. Apers. The miRRor MMDBMS Architecture. In M.P. Atkinson, M.E. Orlowska, P. Valduriez, S.B. Zdonik, and M.L. Brodie, editors, *Proceedings of the 25th VLDB Conference*, pages 758–761. VLDB, Morgan Kaufmann, September 1999.

[43] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 425–436, 2001.

A INEX NEXI queries

A.1 2003 Queries

```
61. //article[about(.,clustering +distributed) and about(.//sec,java)]
62. //article[about(.,security +biometrics) AND about(.//sec,"facial recognition")]
63. //article[about(.,"digital library")
       AND about(.//p, +authorization +"access control" +security)]
64. //article[about(., hollerith)]//sec[about(., dehomag)]
65. //article[.//fm//yr > 1998 AND about(., "image retrieval")]
66. //article[.//fm//yr < 2000]//sec[about(.,"search engines")]
67. //article//fm[about(.//(tig|abs), +software +architecture)
       and about(., -distributed -web)]
68. //article[about(., +smalltalk) or about(., +lisp) or about(.,+erlang)
       or about(., +java)]//bdy//sec[about(., +"garbage collection" +algorithm)]
69. //article//bdy//sec[about(.//st,"information retrieval")]
70. //article[about(.//fm//abs, "information retrieval" "digital libraries")]
71. //article[about(.,formal methods verify correctness aviation systems)]
       //bdy//*[about(.,case study application model checking theorem proving)]
72. //article[about(.//fm//au//aff,united states of america)]
       //bdy//*[about(.,weather forecasting systems)]
73. //article[about(.//st,+comparison) and about(.//bib,"machine learning")]
74. //article[about(., video streaming applications)]
       //sec[about(., media stream synchronization)
       OR about(., stream delivery protocol)]
75. //article[about(., petri net) AND about(.//sec, formal definition)
       AND about(.//sec, algorithm efficiency computation approximation)]
76. //article[(.//fm//yr = 2000 OR .//fm//yr = 1999)
       AND about(., "intelligent transportation system")]
       //sec[about(.,automation +vehicle)]
77. //article[about(.//sec,"reverse engineering")]//sec[about(., legal)
       OR about(.,legislation)]
78. //vt[about(.,"information retrieval" student)]
79. //article[about(.,xml) AND about(.,database)]
80. //article//bdy//sec[about(.,"clock synchronization" "distributed systems")]
81. //article[about(.//p,"multi concurrency control") AND about(.//p, algorithm)
       AND about(.//fm//atl, databases)]
82. //article[about(.,handwriting recognition) AND about(.//fm//au,kim)]
83. //article//fm//abs[about(., "data mining" "frequent itemset")]
84. //p[about(.,overview "distributed query processing" join)]
85. //article[.//fm//yr >= 1998 and .//fig//no > 9]//sec[about(.//p, VR
       "virtual reality" "virtual environment" cyberspace "augmented reality")]
86. //sec[about(.,mobile electronic payment system)]
87. //article[(.//fm//yr = 1998 OR .//fm//yr = 1999 OR .//fm//yr = 2000
       OR .//fm//yr = 2001 OR .//fm//yr = 2002)
       AND about(., "support vector machines")]
88. //article[(.//fm//yr = 1998 OR .//fm//yr = 1999 OR .//fm//yr = 2000
       OR .//fm//yr = 2001) AND about(., "web crawler")]
89. //article[about(.//bdy,clustering "vector quantization" +fuzzy +k-means
       +c-means -sofm -som)]//bm//bb[about(.,"vector quantization" +fuzzy
       clustering +k-means +c-means) AND about(.//pdt,1999)
       AND about(.//au//snm, -kohonen)]
```

```
90. //article[about(.//sec,+trust authentication "electronic commerce" e-commerce
       ebusiness marketplace)]//abs[about(., trust authentication)]
```

A.2 2004 Queries

```
127. //sec//(p|fgc)[about( ., godel lukasiewicz and other fuzzy implication
        definitions)]
128. //article[about(., intelligent transport systems)]//sec[about(., on-board
        route planning navigation system for automobiles)]
129. //article[about(.//atl, new book review bookshelf)]
        //sec[about(., database "data warehouse")]
130. //article[about(.//p,object database)]//p[about(.,version management)]
131. //article[about(.//au,"jiawei han")]//abs[about(.,"data mining")]
132. //article[about(.//abs,classification)]//sec[about(.,experiment compare)]
133. //article[about(.//fm//tig//atl, query) and about(.//st, optimization)]
134. //article[(about(., "phrase search") OR about(., "proximity search")
        OR about(., "string matching")) AND (about(.,tries)
        OR about(.,"suffix trees") OR about(.,"pat arrays"))]
        //sec[about(.,algorithm)]
135. //article[about(.//atl, summaries)]//sec[about(., "Internet security")
        or about(.,"network security")]
136. //bib[about(., text categorisation) and about(., "support vector machines" svm)]
137. //article [about(.//abs, "digital library") or about(.//ip1, "digital library")]
138. //article[about(.,operating system) and about(.//sec,thread implementation)]
139. //article[(about(.//bb//au//snm, bertino) or about(.//bb//au//snm , jajodia))
        and about(.//bb//atl, security model) and about(.//bb//atl, -"indexing model"
        -"object oriented model")]
140. //article[about(., xml)]//bdy//sec[about(., "information integration" +web)
        or about(., "information exchange" +web)]
141. //article[about(.,java)]//sec[about(.,implementing threads)]
142. //abs[about(.,database access using perl)]
143. //sec[about(.,+stemming +information)]
144. //article[about(.//abs, bioinformatics "software systems")
        and about(.//p,data warehouse multi-format multi-type integration)]
145. //article[about(.,information retrieval)]//p[about(.,relevance feedback)]
146. //article[(.//fm//yr > 1999)]//sec[about(.//*, xml html web)]
147. //sec[about(.//st,conclusions) AND about(.,web) AND about(.,distance)
        AND about (.,learning)]
149. //article[about(.//(abs|kwd), "genetic algorithm")]
        //bdy//sec[about(.,simulated annealing)]
150. //article[about(., animation)] //bdy//sec[about(.//st, inverse kinematics)]
151. //article[about(., "web search engine")]//sec[about(., "vector space model")]
152. //article//p[about(.,+linux "word processing" "word processor"
        "office programs")]
153. //article//bm//vt[about(.,"phd student") OR about(.,"phd candidate")]
154. //article[about(.//bib, abiteboul)]//bdy//*[about(., semistructured + query)]
155. //article[about(.//p,"self organising feature map") and about(.//fm//yr,2000)]
        //fig[about(.//fgc,"self organising map")]
156. //article[about(.//abs, "spatial join")]
        //bdy//sec[about(., "performance evaluation")]
```

```
157. //article[about(.//abs,-query -"query optimization" -linear)
        and about(.//bdy,newton +gradient hessian technique)]
        //bdy//*[about(.,+optimization -experiments)
        and (about(.,maximization) or about(.,minimization))]
158. //article[about(.//fm, turing test) or about(.//abs, turing test)]
        //bdy[about(., turning test consciousness intelligence imitation game)]
159. //article[about(.,"bayesian networks")]//sec[about(.,learning structure)]
160. //article[about(., image retrieval)]
        //sec[about(., "latent semantic indexing")]
161. //article[about(., database access methods for spatial data and text)]
        //bm//bb[about(.//atl, database access methods)]
```

B Execution times

The hardware platform used to produce the results presented in this section is an old PC running Linux 2.4.20-8smp, with two Pentium™ III 600 MHz CPUs, 1 GB of main-memory, and a 100 GB disk array mounted in RAID 0 (striping) mode.

Here we give avarage execution times per topic on INEX 2003 topics for illustration. We did not tune our system for efficiency and the prototype implementation was a straightforward one. We give some of the execution times for comparison among different models. Note that for most of the topics the execution time is less than double the time stated as average time per query specified in Table 4. This is due to the fact that in both topic sets (INEX 2003 and 2004) there are few queries that use `//*` expression, which is in our system implemented as a selection of all containing regions, that consume vaste amount of memory and whose execution is very slow. Such topics are for example topics 71 and 72 from INEX 2003 topic set, and topics 146, 154, and 157 from INEX 2004 topic set.

Table 4: Execution times for some of the 2003 runs averaged per query

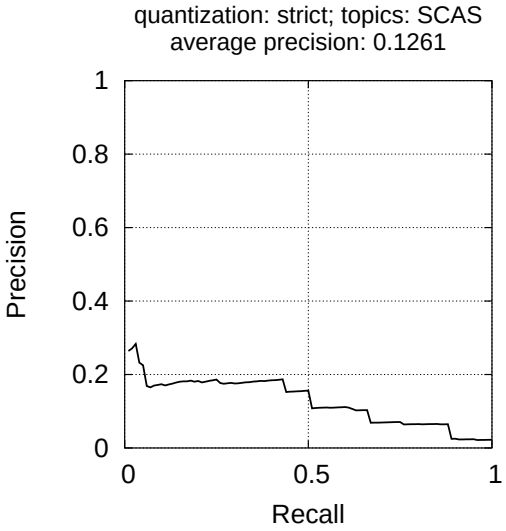
Model	**Execution time (seconds)**
LM_nomod_nophr_sum_prod_wsa_sum_true_no_ent	135.5
LMs_nomod_nophr_sum_prod_wsa_sum_true_no_ent_05	149.9
LMs_nomod_nophr_sum_sum_wsa_sum_true_no_ent_05	149.4
LMs_nomod_nophr_sum_prod_sum_sum_true_no_ent_05	133.9
GPX_nomod_nophr_sum_exp_sum_sum_true_no_ent_5	59.9
GPX_nomod_nophr_exp_exp_sum_sum_true_no_ent_5	60.1
Okapi_nomod_nophr_prob_prod_wsa_sum_true_no_ent_15_075	245.3
Okapi_nomod_nophr_sum_sum_wsa_sum_true_no_ent_15_075	246.9
tfidf_nomod_nophr_sum_sum_wsa_sum_true_no_ent	198.5
tfidf_nomod_nophr_max_min_wsa_sum_true_no_ent	201.3
tfidf_nomod_nophr_sum_prod_wsa_sum_true_no_ent	197.9
tfidf_nomod_nophr_prob_prod_wsa_sum_true_no_ent	191.0

C Experimental results

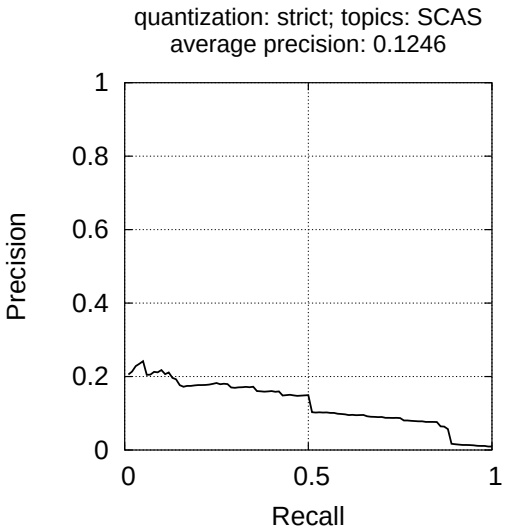
C.1 Plain Runs for 2003 Queries

C.1.1 Language models without smoothing

INEX 2003: LM_nomod_nophr_sum_prod_wsa_sum_true_no_ent

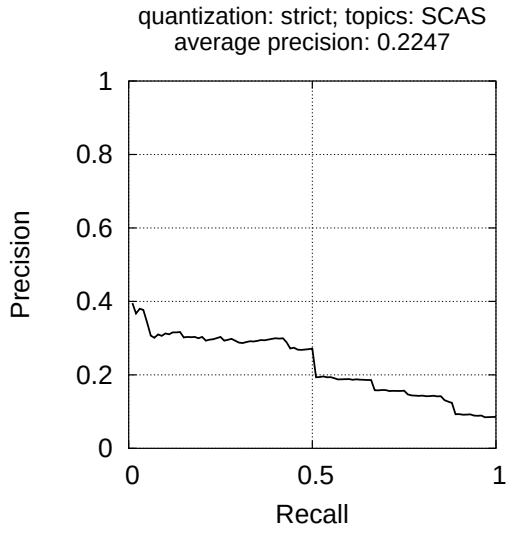


INEX 2003: LM_nomod_nophr_sum_sum_wsa_sum_true_no_ent

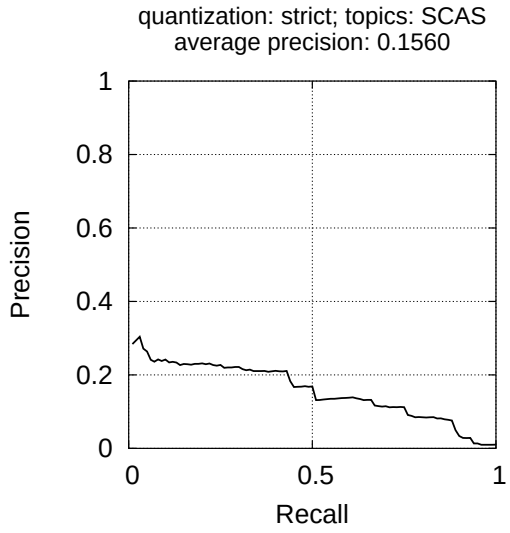


C.1.2 Language models with smoothing

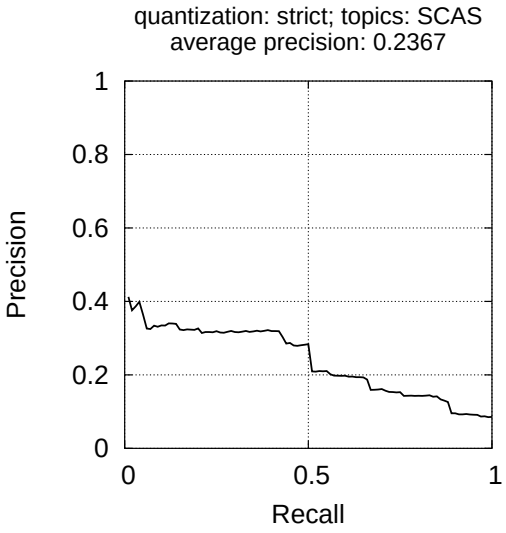
INEX 2003: LMs_nomod_nophr_sum_prod_wsa_sum_true_no_ent_05



INEX 2003: LMs_nomod_nophr_sum_sum_wsa_sum_true_no_ent_05

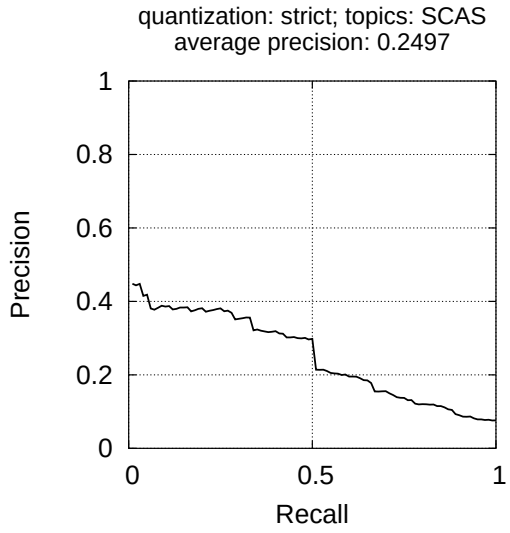


INEX 2003: LMs_nomod_nophr_sum_prod_sum_sum_true_no_ent_05

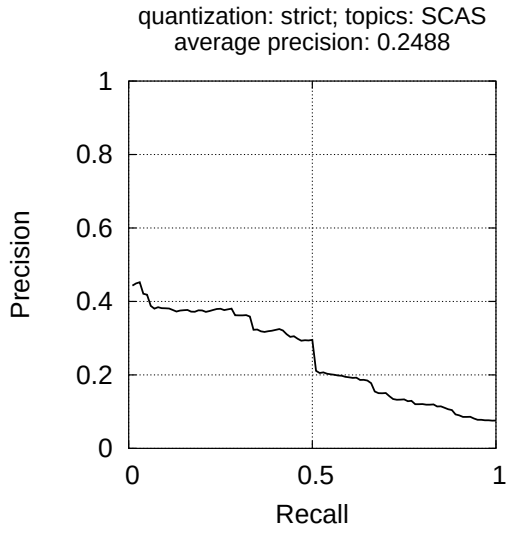


C.1.3 Language models with document weighting

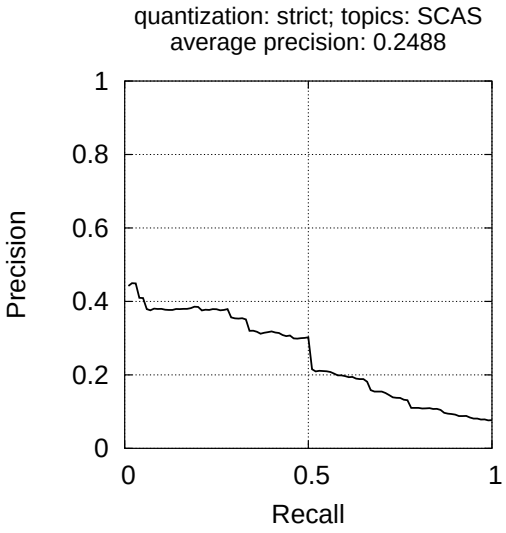
INEX 2003: LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_01_05



INEX 2003: 3LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_01_03

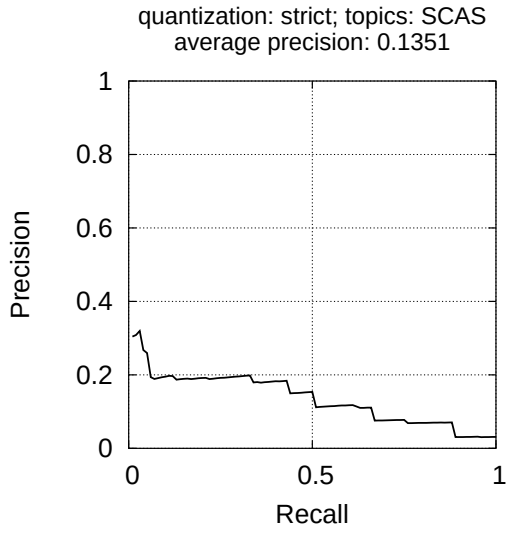


INEX 2003: 3LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_02_06

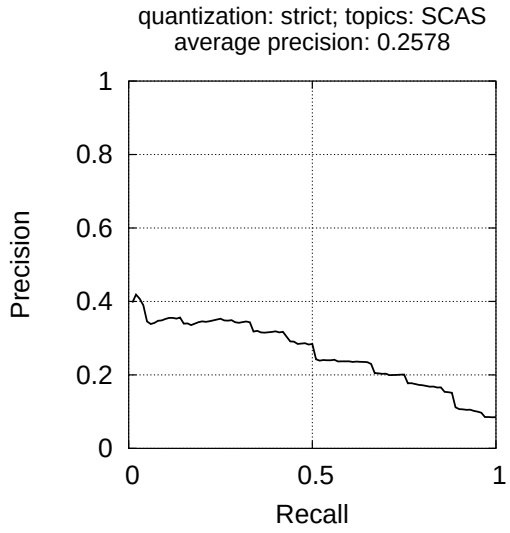


C.1.4 Okapi

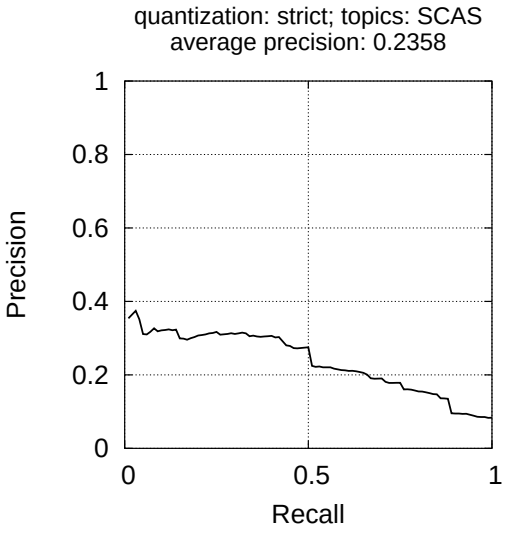
INEX 2003: Okapi_nomod_nophr_prob_prod_wsa_sum_true_no_ent_15_075



INEX 2003: Okapi_nomod_nophr_sum_sum_wsa_sum_true_no_ent_15_075

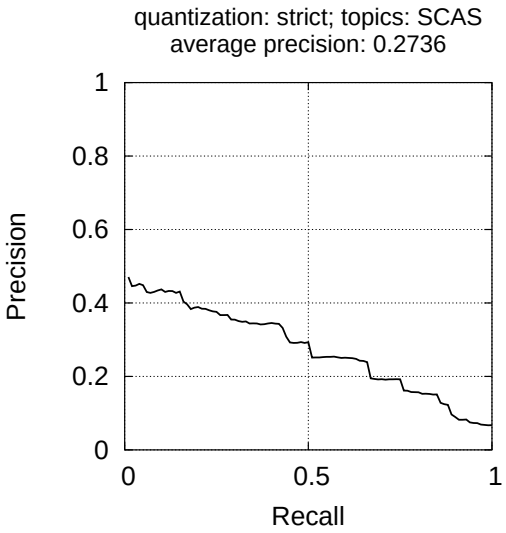


INEX 2003: Okapi_nomod_nophr_sum_sum_sum_sum_true_no_ent_15_075

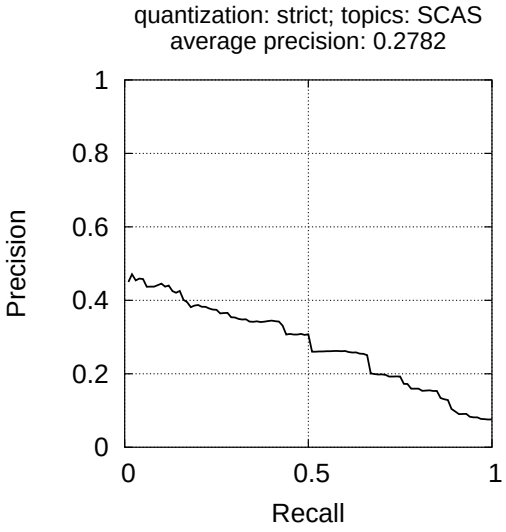


C.1.5 GPX

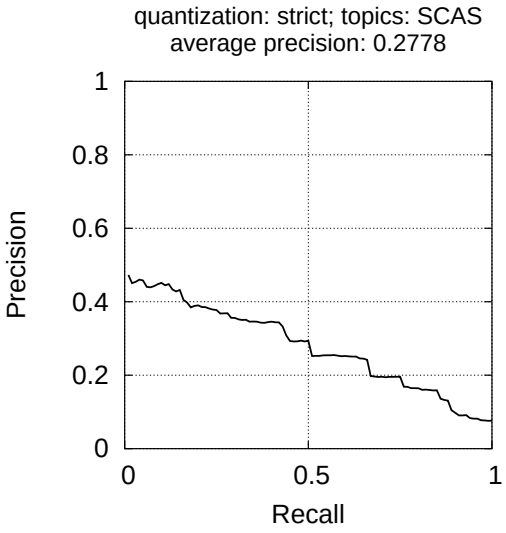
INEX 2003: GPX_nomod_nophr_sum_exp_sum_sum_true_no_ent_5



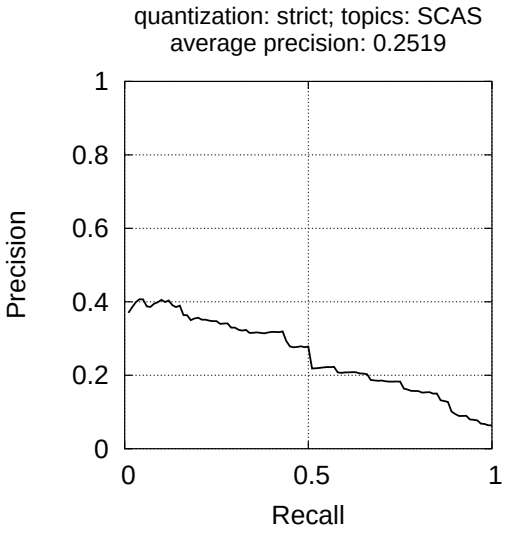
INEX 2003: GPX_nomod_nophr_exp_exp_sum_sum_true_no_ent_5_1



INEX 2003: GPX_nomod_nophr_exp_exp_sum_sum_true_no_ent_5

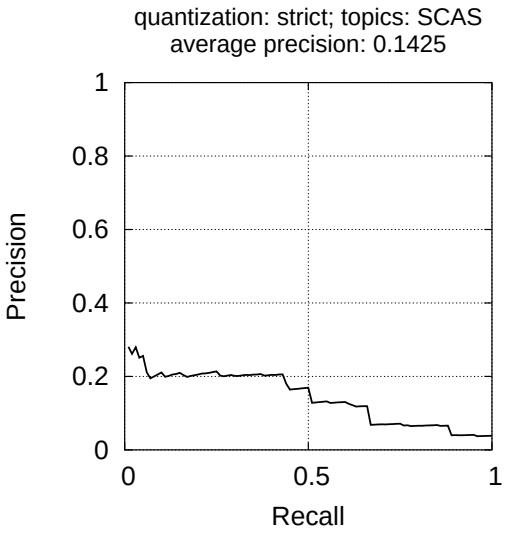


INEX 2003: GPX_nomod_nophr_exp_exp_wsa_sum_true_no_ent_5

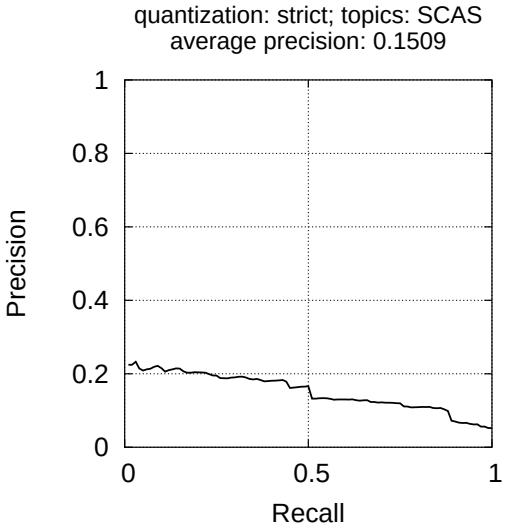


C.1.6 tf.idf

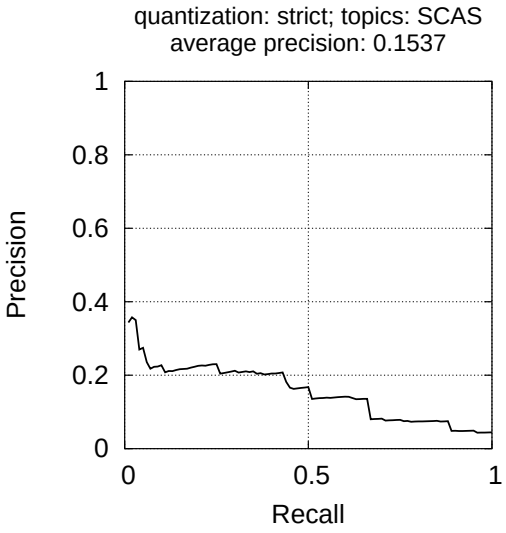
INEX 2003: tfidf_nomod_nophr_max_min_wsa_sum_true_no_ent



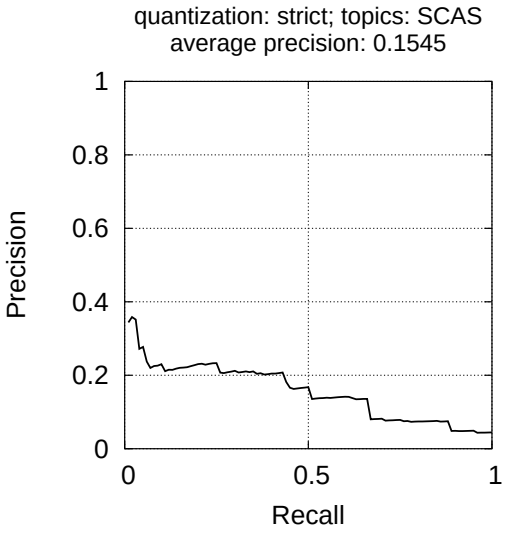
INEX 2003: tfidf_nomod_nophr_sum_sum_wsa_sum_true_no_ent



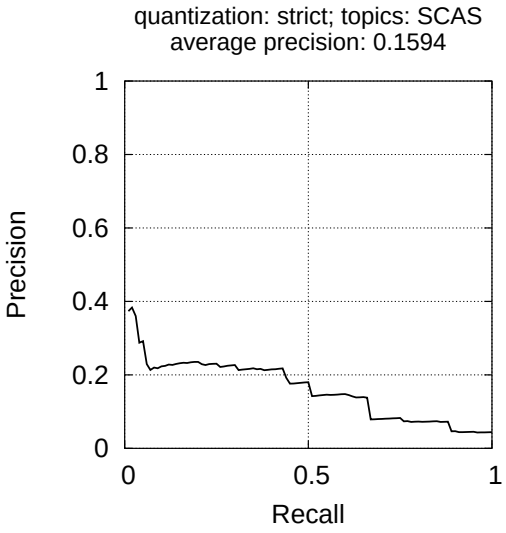
INEX 2003: tfidf_nomod_nophr_prob_prod_wsa_sum_true_no_ent



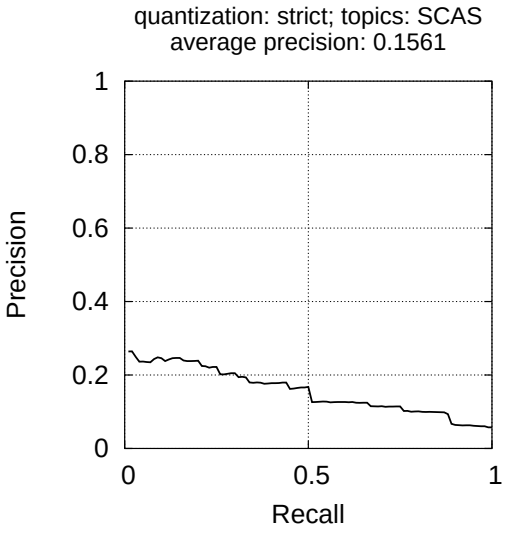
INEX 2003: tfidf_nomod_nophr_sum_prod_wsa_sum_true_no_ent



INEX 2003: tfidf_nomod_nophr_prob_prod_sum_sum_true_no_ent



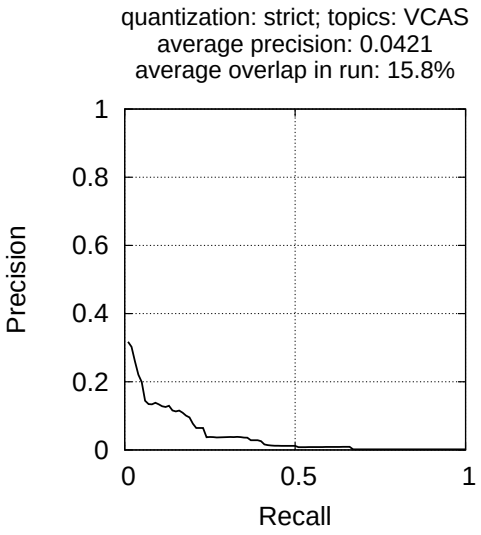
INEX 2003: tfidf_nomod_nophr_sum_sum_sum_sum_true_no_ent



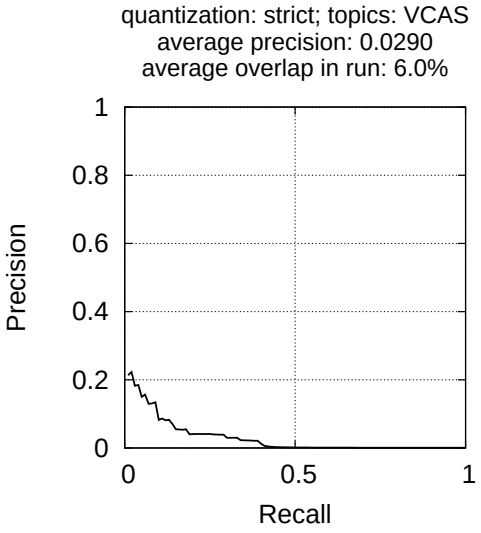
C.2 Plain Runs for 2004 Queries

C.2.1 Language models without smoothing

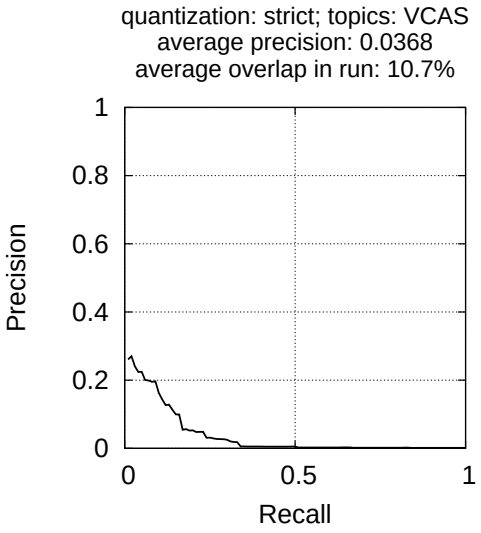
INEX 2004: Bool_nomod_nophr_sum_sum_wsa_sum_true_no_ent



INEX 2004: LM_nomod_nophr_sum_prod_wsa_sum_true_no_ent

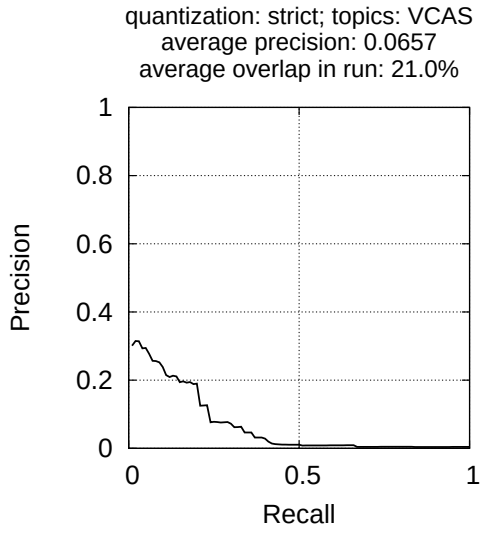


INEX 2004: LM_nomod_nophr_sum_sum_wsa_sum_true_no_ent

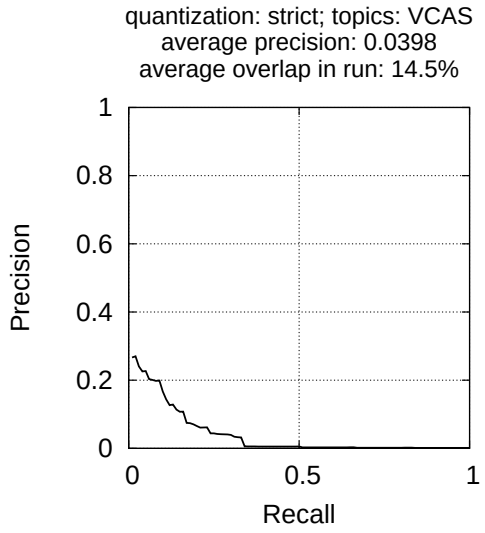


C.2.2 Language models with smoothing

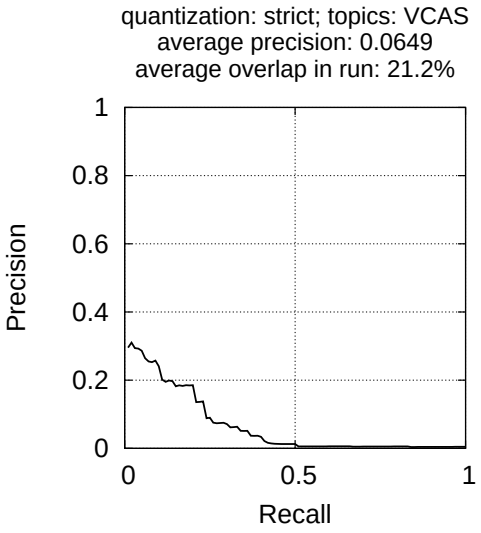
INEX 2004: LMs_nomod_nophr_sum_prod_wsa_sum_true_no_ent_05



INEX 2004: LMs_nomod_nophr_sum_sum_wsa_sum_true_no_ent_05

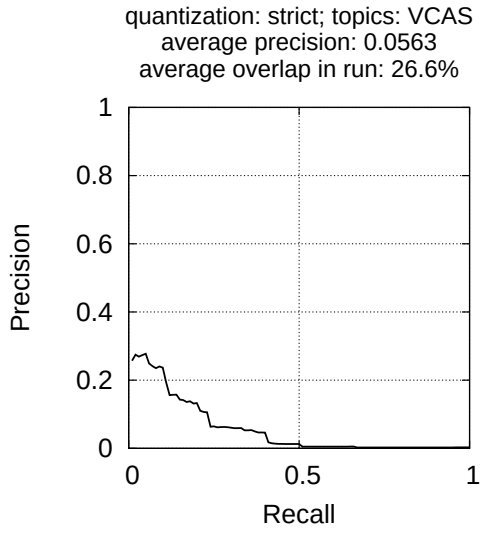


INEX 2004: LMs_nomod_nophr_sum_prod_sum_sum_true_no_ent_05

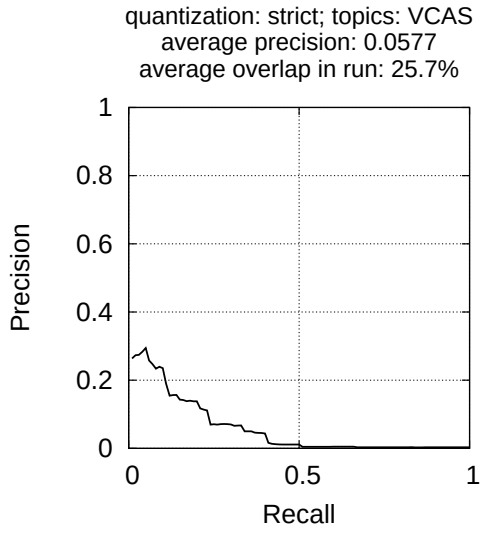


C.2.3 Language models with document weighting

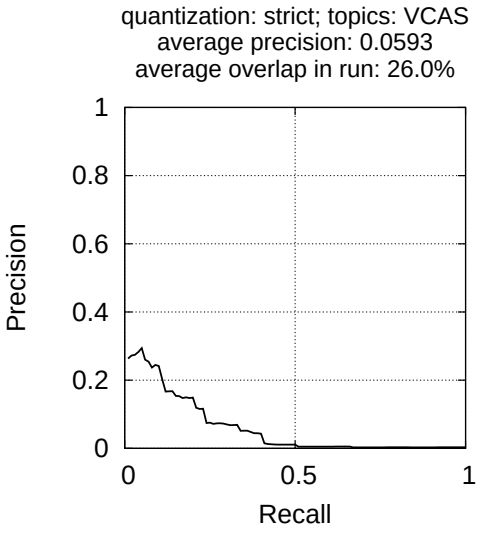
INEX 2004: LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_01_05



INEX 2004: LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_01_03

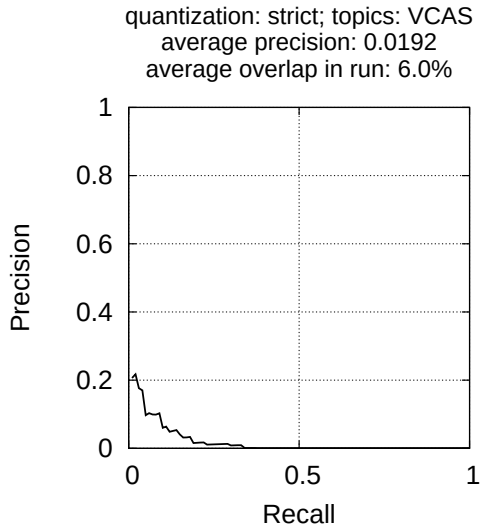


INEX 2004: LMA_nomod_nophr_sum_prod_sum_sum_true_no_ent_02_06

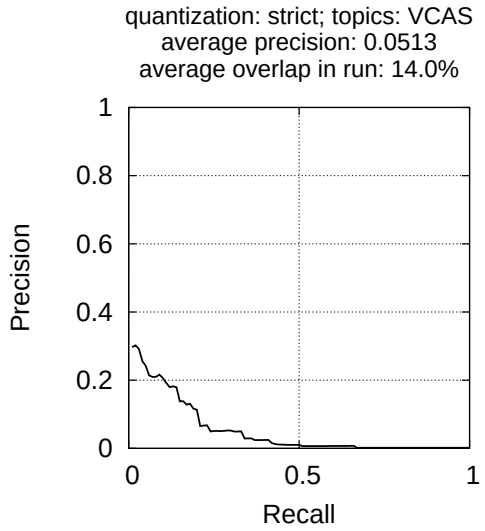


C.2.4 Okapi

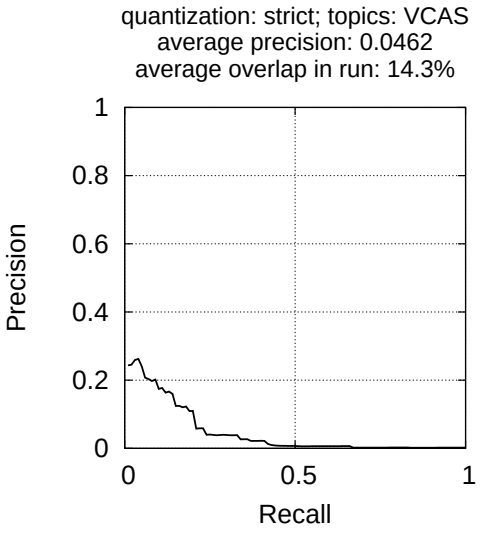
INEX 2004: Okapi_nomod_nophr_prob_prod_wsa_sum_true_no_ent_15_075



INEX 2004: Okapi_nomod_nophr_sum_sum_wsa_sum_true_no_ent_15_075

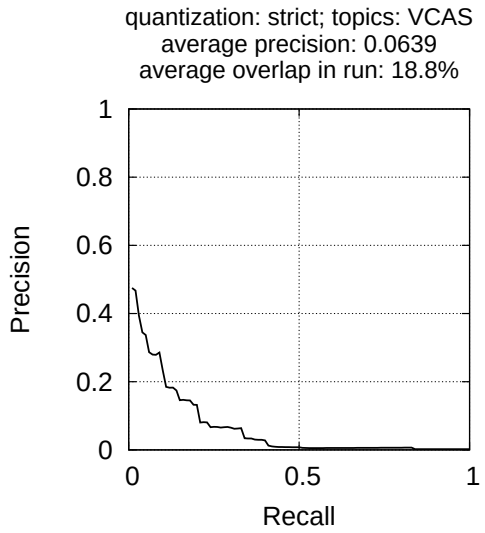


INEX 2004: Okapi_nomod_nophr_sum_sum_sum_sum_true_no_ent_15_075

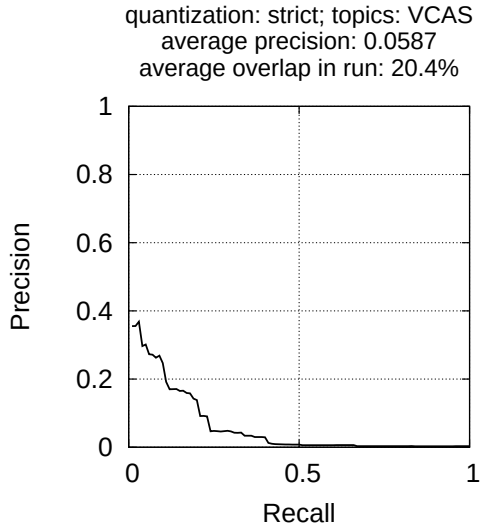


C.2.5 GPX

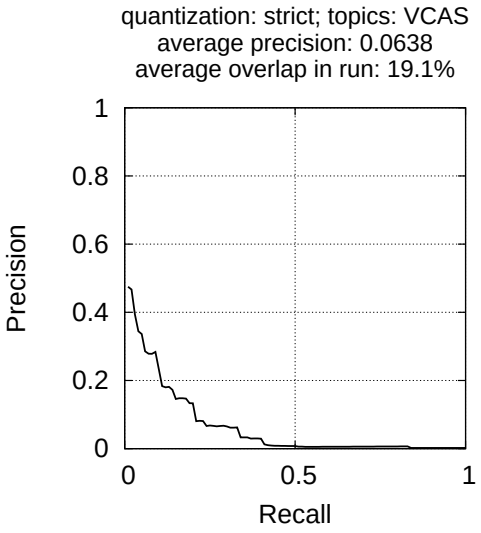
INEX 2004: GPX_nomod_nophr_sum_exp_sum_sum_true_no_ent_5



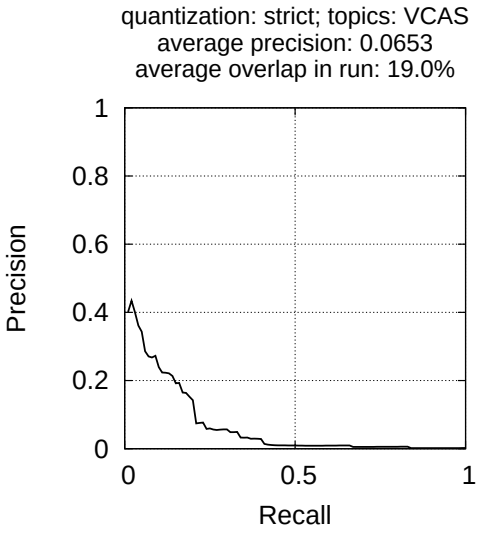
INEX 2004: GPX_nomod_nophr_exp_exp_sum_sum_true_no_ent_5_1



INEX 2004: GPX_nomod_nophr_exp_exp_sum_sum_true_no_ent_5

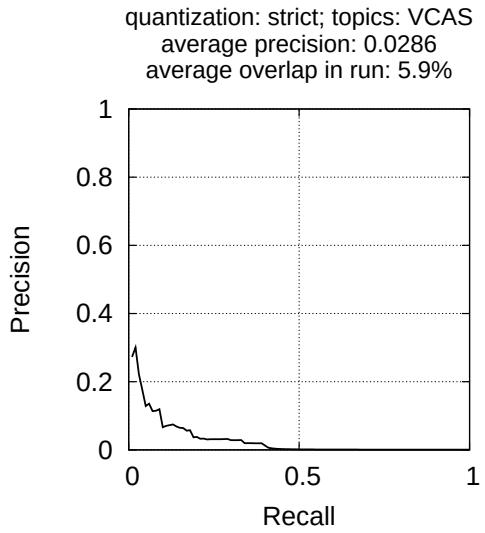


INEX 2004: GPX_nomod_nophr_exp_exp_wsa_sum_true_no_ent_5

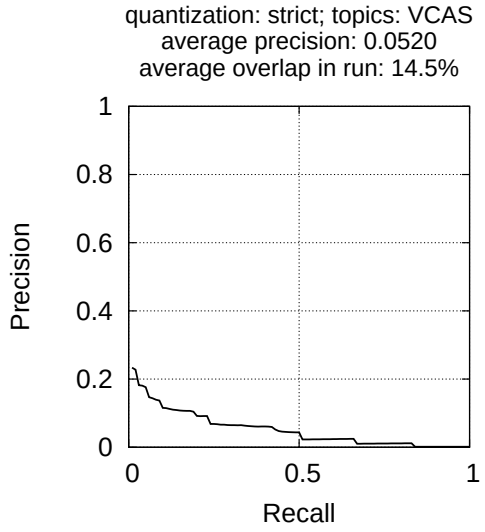


C.2.6 tf.idf

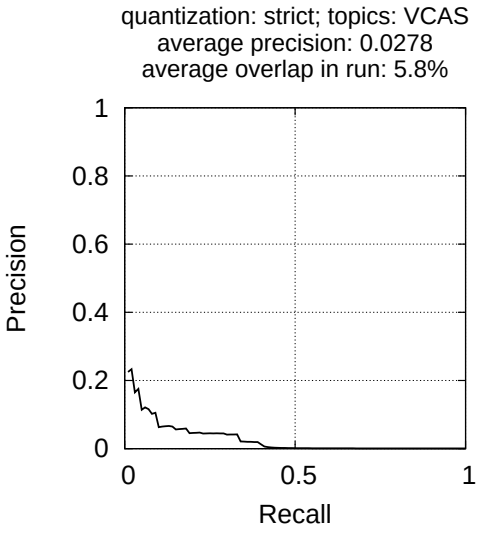
INEX 2004: tfidf_nomod_nophr_max_min_wsa_sum_true_no_ent



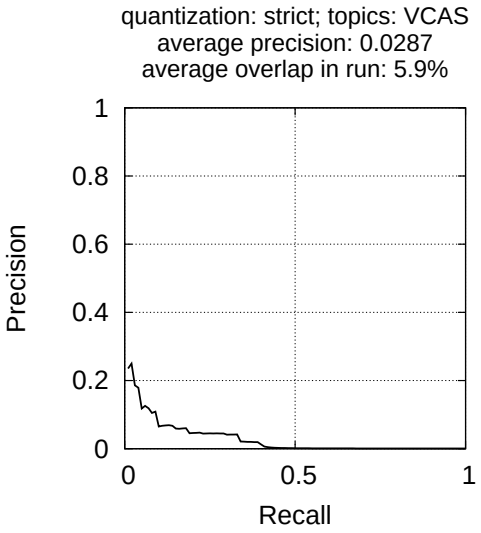
INEX 2004: tfidf_nomod_nophr_sum_sum_wsa_sum_true_no_ent



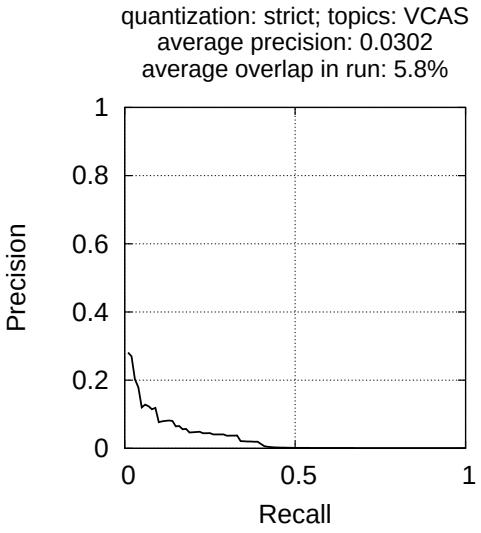
INEX 2004: tfidf_nomod_nophr_prob_prod_wsa_sum_true_no_ent



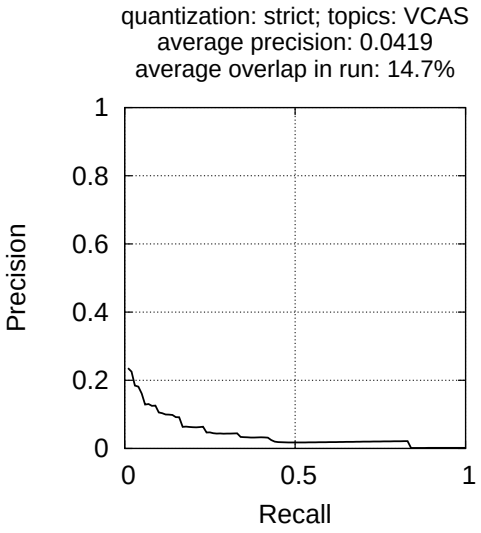
INEX 2004: tfidf_nomod_nophr_sum_prod_wsa_sum_true_no_ent



INEX 2004: tfidf_nomod_nophr_prob_prod_sum_sum_true_no_ent



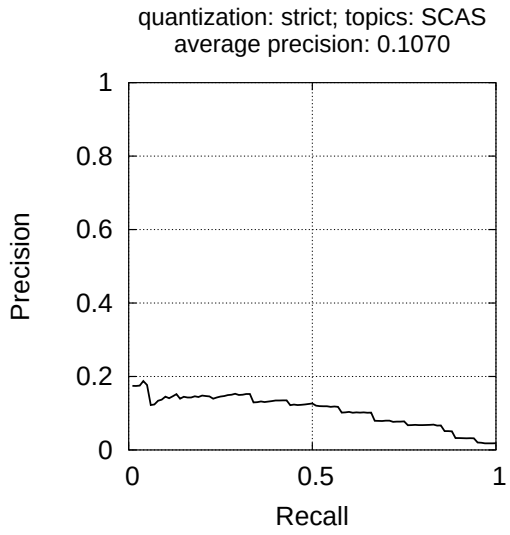
INEX 2004: tfidf_nomod_nophr_sum_sum_sum_sum_true_no_ent



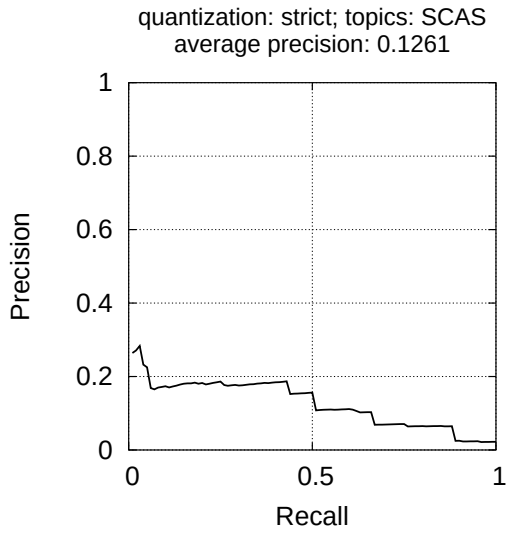
C.3 Runs with Modifiers and Scaling on 2003 Queries

C.3.1 Boolean-like and language models without smoothing

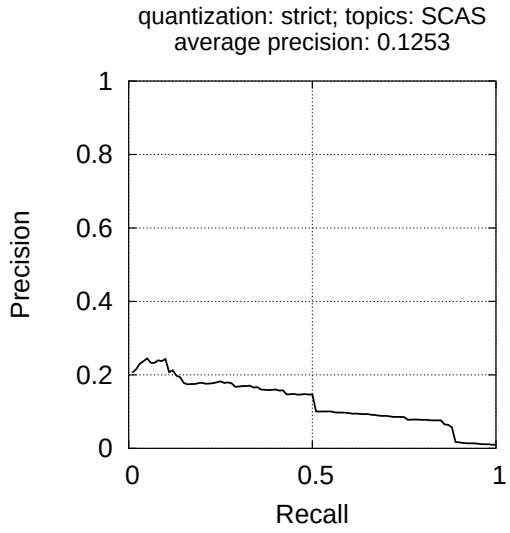
INEX 2003: Bool_mod_nophr_sum_sum_wsa_sum_true_no_ent



INEX 2003: LM_mod_nophr_sum_prod_wsa_sum_true_no_ent

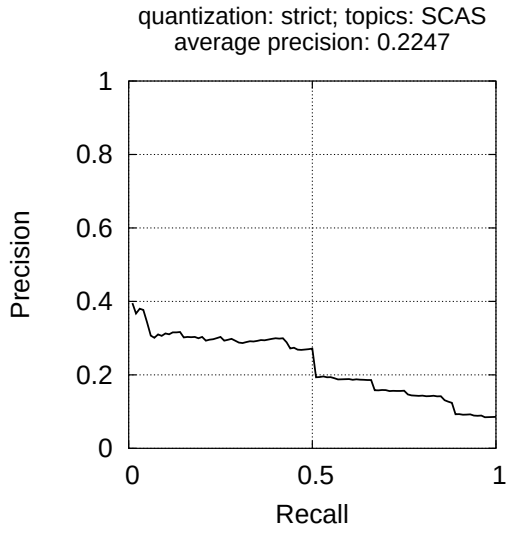


INEX 2003: LM_mod_nophr_sum_sum_wsa_sum_true_no_ent

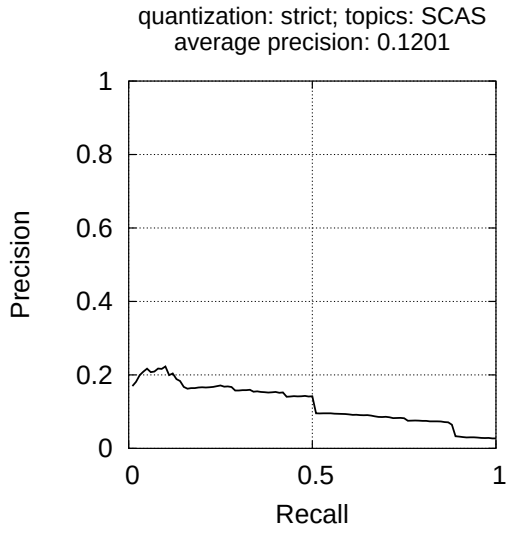


C.3.2 Language models with smoothing

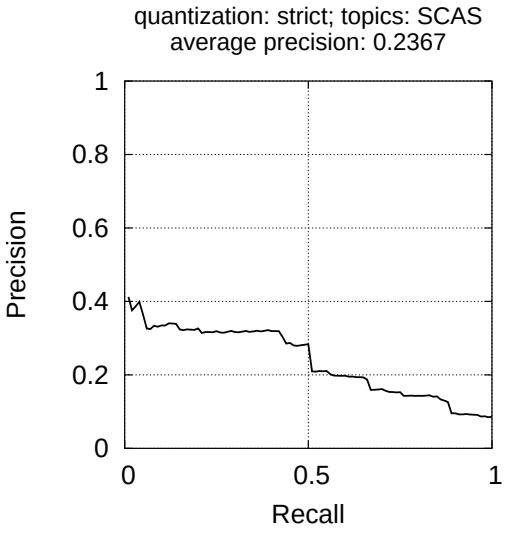
INEX 2003: LMs_mod_nophr_sum_prod_wsa_sum_true_no_ent_05



INEX 2003: LMs_mod_nophr_sum_sum_wsa_sum_true_no_ent_05

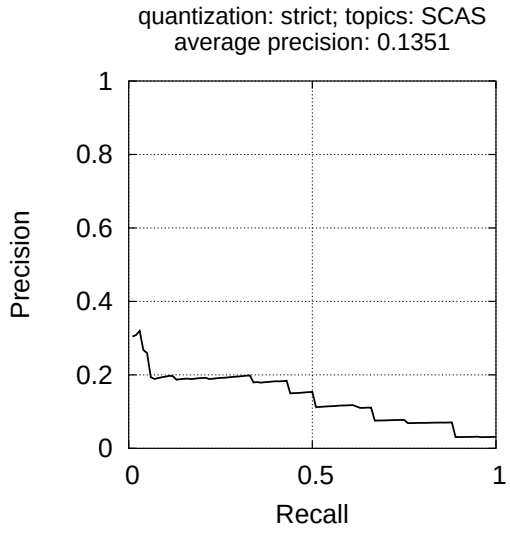


INEX 2003: LMs_mod_nophr_sum_prod_sum_sum_true_no_ent_05

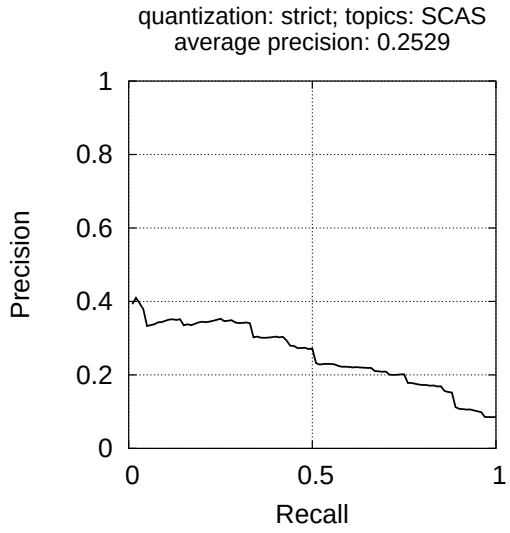


C.3.3 Okapi

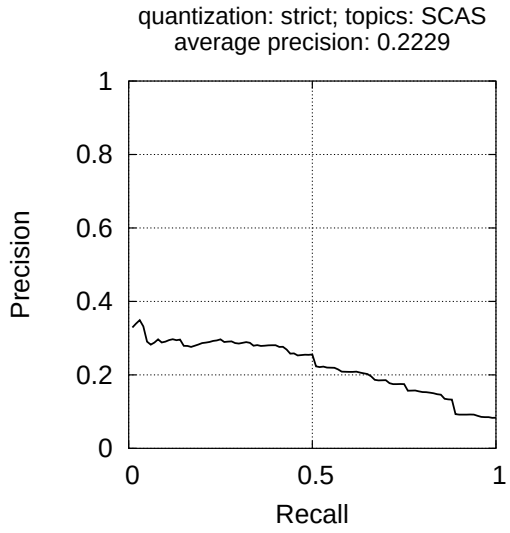
INEX 2003: Okapi_mod_nophr_prob_prod_wsa_sum_true_no_ent_15_075



INEX 2003: Okapi_mod_nophr_sum_sum_wsa_sum_true_no_ent_15_075

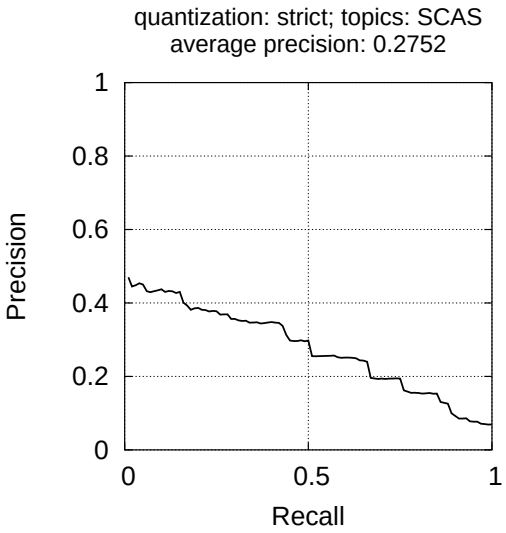


INEX 2003: Okapi_mod_nophr_sum_sum_sum_sum_true_no_ent_15_075

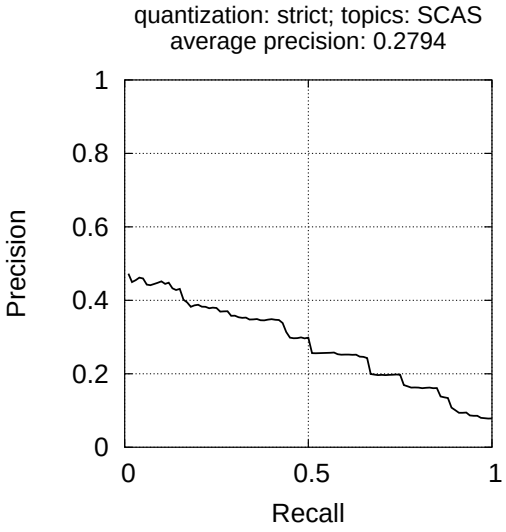


C.3.4 GPX

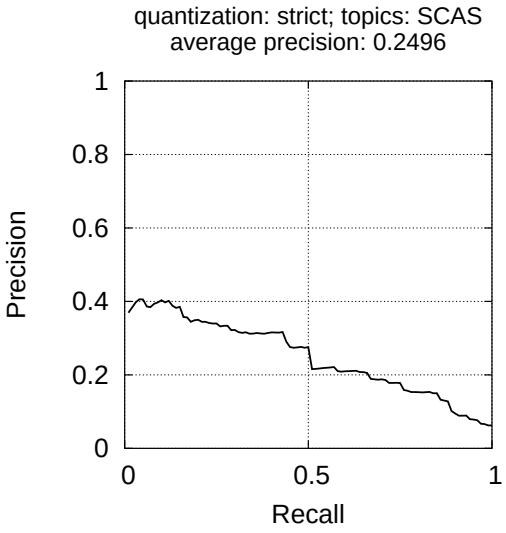
INEX 2003: GPX_mod_nophr_sum_exp_sum_sum_true_no_ent_5



INEX 2003: GPX_mod_nophr_exp_exp_sum_sum_true_no_ent_5

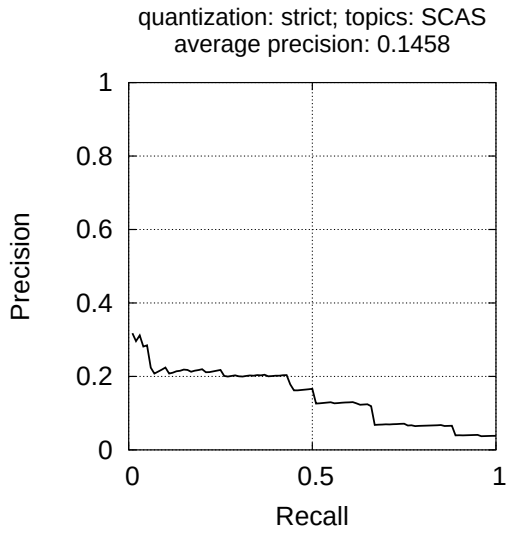


INEX 2003: GPX_mod_nophr_exp_exp_wsa_sum_true_no_ent_5

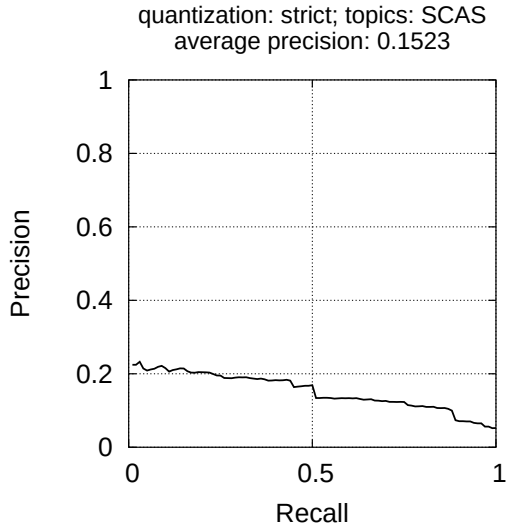


C.3.5 tf.idf

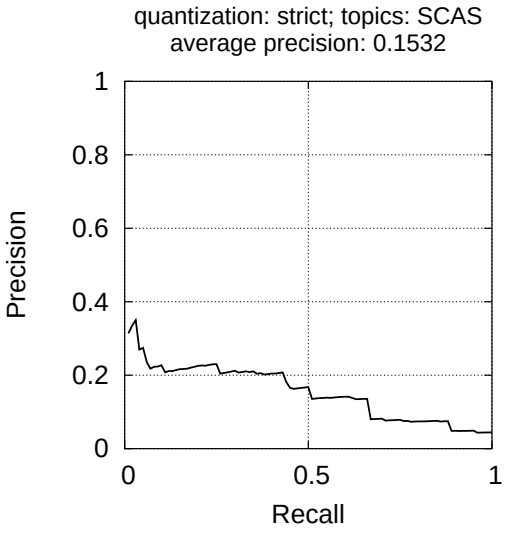
INEX 2003: tfidf_mod_nophr_max_min_wsa_sum_true_no_ent



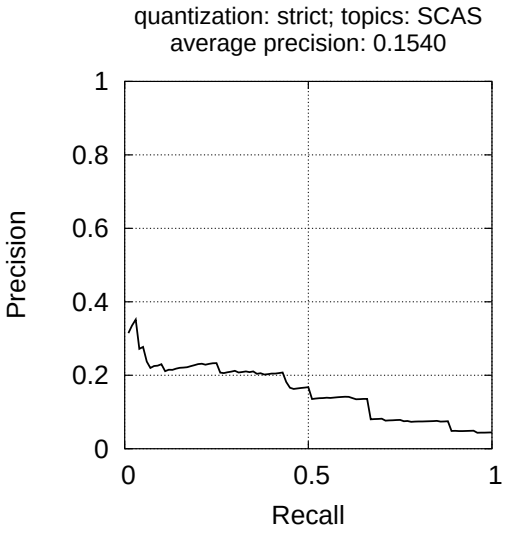
INEX 2003: tfidf_mod_nophr_sum_sum_wsa_sum_true_no_ent



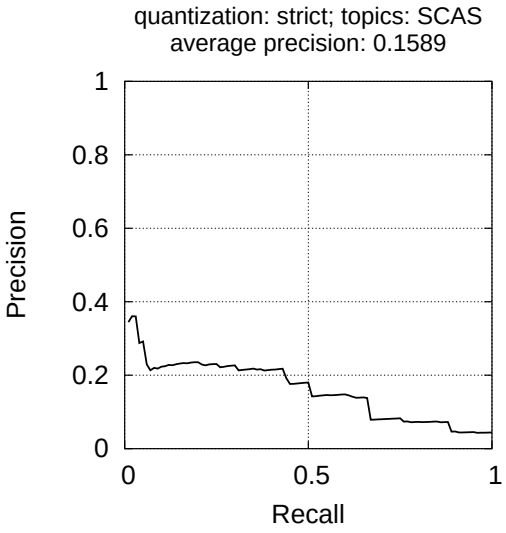
INEX 2003: tfidf_mod_nophr_prob_prod_wsa_sum_true_no_ent



INEX 2003: tfidf_mod_nophr_sum_prod_wsa_sum_true_no_ent



INEX 2003: tfidf_mod_nophr_prob_prod_sum_sum_true_no_ent



INEX 2003: tfidf_nomod_nophr_sum_sum_sum_sum_true_no_ent