Exploiting Query Structure and Document Structure to Improve Document Retrieval Effectiveness

Vojkan Mihajlović Djoerd Hiemstra Henk Ernst Blok 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

In this paper we present a systematic analysis of document retrieval using unstructured and structured queries within the *score region algebra* (*SRA*) structured retrieval framework. The behavior of different retrieval models, namely Boolean, tf.idf, GPX, language models, and Okapi, is tested using the transparent SRA framework in our three-level structured retrieval system called *TIJAH*. The retrieval models are implemented along four elementary retrieval aspects: element and term selection, element score computation, score combination, and score propagation.

The analysis is performed on a numerous experiments evaluated on TREC and CLEF collections, using manually generated unstructured and structured queries. Unstructured queries range from the short title queries to long title + description + narrative queries. For generating structured queries we exploit the knowledge of the document structure and the content used to semantically describe or classify documents. We show that such structured information can be utilized in retrieval engines to give more precise answers to user queries then when using unstructured queries.

Categories and Subject Descriptors: H.3.3 [Information search and retrieval]: Query formulation, Retrieval models; H.2.3 [Languages]: Query languages;

General Terms: languages, experimentation

Keywords: structured queries, region algebra, retrieval models, structured retrieval, retrieval effectiveness

1. INTRODUCTION

In the times of lazy users and the majority of the documents on the Web lacking clear structure and uniform format, the effectiveness of information retrieval engines seems to reach the top. The question is whether the upcoming years will bring significant improvements for IR engines that will be appreciated by users.

Although most of the user queries issued to Web search engines constitute of a limited number of query terms (93% of the Web queries have less than five terms in the query as reported in [19]), these query terms do not have to be the only input that the search engines can use to find the relevant information. For instance, faceted queries [12, 18, 35] utilize knowledge about the semantic relations between words to retrieve more relevant answers, whereas field (structured) search queries utilize document structure to give more precise answers (see [14]).

What is not well utilized in many text retrieval approaches so far is that many of the "flat-file" documents are actually more/less structured using one of the following formats: HTML, SGML, XML, etc. Additionally, some documents come with an already prepared classification or a short description of document content (e.g., keywords) in some of the tags. For example, TREC and CLEF collections are in SGML format with tags such as: "subject", "section", "type". Thus, the semantics of these tags and the annotation of the documents that they contain can also be utilized for improving the effectiveness of a retrieval system.

We argue that the structured organization of documents as well as the structural formation of queries should not be neglected in future information retrieval engines. This idea is addressed in a number of research papers, starting with the early work by Callan, Turtle, and Croft [7, 11] on incorporating hypertext links and field-based queries into retrieval systems, and Salminen and Tompa [32] and Burkowski and Clarke [5, 9] on structured text search. These papers showed the usefulness of structured retrieval and set the path for future research.

Furthermore, the importance of field search [6, 10, 37] and structural (XML) search [1, 13, 22, 24, 28] is also realized by many commercial search engines nowadays. For instance, Google has "advanced search" that supports a number of field search extensions, such as domain search (*site:*), file type search (*filetype:*), and URL search (*inurl:*)[1], and the U.S. National Center for Biotechnology Information uses a search engine that searches various medical data sources based on their structure, e.g., users can search the PubMed medical collection based on *subsets* (e.g., AIDS, cancer), *ages* (e.g., infant, child, adult)[2], etc. Additionally, there are open source search engines like Indri [37] and Terrier [29], or enterprise search engines like Panoptic [16], that enable the formulation of complex structured queries and their evaluation.

[1] http://www.google.com/.
[2] http://www.ncbi.nlm.nih.gov/.

The growth of personalized search, i.e., the information about the user who is performing the search [23, 30], as well as the semantic information that can be deduced from the search request, e.g., using semantic web resources [36], enables the automatic generation of such complex queries. Therefore, in the future we can expect that the automatic generation of structured queries would be possible, or at least that many retrieval systems will support users in expressing their information need as a structured query. In this paper we assume that the generation of complex (structured) queries is possible and focus on analyzing the system performance having such queries.

Not many systematic analysis that explore the usage of document structure for document retrieval have been presented in the literature. For such analysis, it is beneficial to have a framework that enables the implementation of various retrieval models, and that supports retrieval from structured documents using unstructured and structured queries. Such framework would help us in improving the effectiveness of existing document retrieval approaches.

1.1 Research focus

Much research has been done into retrieving relevant documents given a simple list-of-term queries. Structured retrieval is gaining more and more attention nowadays. We argue that the automatic generation of simple structured queries, such as faceted and field search queries explained in Section 2.2, is by far more realistic than the generation of long queries such as TREC title + description or title + description + narrative. However, it is unclear whether we can achieve comparable effectiveness when using these 'simple' structured queries in comparison with expanded queries (using topic description and narrative).

The question that we try to answer in this paper is twofold. The first part is related to the analysis of the behavior of different retrieval models in unstructured and structured retrieval scenarios. The second part tests whether the formation of structured queries by (end) users, either with or without the help of information automatically extracted from documents, can improve retrieval model effectiveness. In other words, is a search system that is provided with the advanced structured queries (either faceted, field-based, or both) more effective on ad-hoc search task than the same search engine operating on simple or expanded unstructured text queries.

1.2 Research methodology

To test the robustness of different retrieval models with respect to different unstructured and structured query formulations, we perform the experiments using the TIJAH system [26]. TIJAH is a transparent three-level database system. Its central part is *score region algebra* (*SRA*) at the logical level. SRA supports a variety of retrieval models and easy formulation of unstructured and structured queries. An important feature of SRA is that it is not bound to a predefined retrieval model but it supports arbitrary retrieval models. The analysis of the retrieval model behavior for different query specifications is performed along four elementary retrieval aspects [24]: element and term selection, element score computation, score combination, and score propagation. Each aspect is implemented in score region algebra using one or more operators.

We choose three directions for exploring the potential usage of query structure to test whether we can make the retrieval more effective in comparison to the list-of-term queries (ranging from short title queries to long title + description + narrative queries):

- (re)formulating the queries using *faceted* (Boolean) query formulation
- (re)formulating the queries using the document structure and the classification already present in structured documents
- combing the queries that utilize document structure with the simple title and faceted queries.

1.3 Outline

The following section explains different query types used in the paper for analyzing unstructured and structured retrieval. Our three-level structured IR database system TIJAH is presented in Section 3, along with the instantiation of different retrieval models in score region algebra. Section 4 describes the test collections we use and the structured query generation process. Experiments aimed at analyzing different retrieval models in unstructured and structured query scenarios, and at exploring whether structured search can improve the effectiveness with respect to list-of-term queries, are discussed in Section 5. Finally, in Section 6 we conclude the paper.

2. QUERY TYPES

This section explains in more detail the query types we analyze in this paper. As we use TREC ad-hoc topic 305 for illustration throughout the paper, here we give the topic title, description, and narrative:

```
   TITLE: most dangerous vehicles
   DESCRIPTION: Which are the most crashworthy, and
least crashworthy, passenger vehicles?
   NARRATIVE: A relevant document will contain
information on the crashworthiness of a given vehicle
or vehicles that can be used to draw a comparison with
other vehicles. The document will have to
describe/compare vehicles, not drivers.  For instance,
it should be expected that vehicles preferred by 16-25
year-olds would be involved in more crashes, because
that age group is involved in more crashes.  I would
view number of  fatalities per 100 crashes to be more
revealing of  a vehicle's crashworthiness than the
number of crashes per 100,000 miles, for example.
```

2.1 Unstructured queries

In the analysis of unstructured querying we use four types of unstructured queries. These are: title (**T**), title + description (**TD**), title + description + narrative (**TDN**), and expanded queries (**E**). Title, title + description, and title + description + narrative queries can easily be formed using TREC and CLEF topic specifications (see the example TREC ad-hoc topic 305 above).

On the other hand, expanded queries are usually formed by manually or automatically extending the topic title with more query terms. If we assume that the user is willing to trade his time/effort in stating his query using more query terms for effectiveness, or an automatic way to expand the query exists, e.g., using WordNet [27], routing [4], or relevance feedback [33], a simple title of the TREC ad-hoc

query 305 can be transformed into a non-structured query that looks like:

```
vehicles crash cars crashworthy death danger
```

In our approach we use manual query expansion (see [12]) based on a faceted query formulation. Section 5.1 explains in more details the effects of manual query expansion, as well as the consequence of using the topic description and narrative in the query formulation, on retrieval effectiveness.

2.2 Structured queries

As already stated in the Introduction, two types of structured queries are investigated in this paper: faceted (Boolean) queries and field-based structured queries. They are explained below.

2.2.1 Faceted query formulation

The origin of this type of Boolean query formulation can be found in the generation of faceted queries [12, 17] that librarians used in early days of information retrieval, even before the computer era. However, not many searchers use and not many retrieval systems support this kind of query formulation. We try to explore whether faceted search should be supported in retrieval systems, i.e., if it is more effective than list-of-term queries.

The generation of faceted queries is based upon the selection of terms that describe the user information need and on the classification of these terms. The classification reflects the distinction between terms that express the same *facet*. Query terms that belong to the same facet are expressed using disjunction (OR combination) on terms, and facets are combined using conjunction (AND combination). For example, if we apply the faceted query paradigm to (expanded) TREC query 305, the query becomes:

```
(vehicles OR cars) AND
     (crash OR crashworthy OR death OR danger)
```

So far not many explicit proofs exist that this kind of query formulation is beneficial for query evaluation, except for some results presented in [18]. However, there are numerous techniques that use the same idea, except that the faceted query formulation is hidden in the layer of query routing and refining, starting with the work of Buckley et al. [4] and Xu and Croft [39]. Additionally, not many models are tested for their robustness with respect to faceted query formulations. We elaborate more on this issue in Sections 4.2 and 5.2.

2.2.2 Using document structure for query formulation

With the rapid growth of the number of XML (structured) documents, structured information retrieval has become a requirement for modern retrieval models (see [14]). Although XML query languages [1, 13, 38] are quite expressive, even the "professional searchers" seem to have problems with formulating good structured queries that contain nested elements and contextual hints [20, 21]. However, a clear indication exists that in XML retrieval structured queries do help early precision (see [21] and other papers in [14]). Kamps et al. [21] also point out that, except for the simple structural query formulation, structure is mostly used as a hint in the query and it is never an inherent part of the information need. Following their reasoning we explore whether and to what extent "structural search hints" in the form of a (combination of) field search(es) can improve retrieval effectiveness.

Structured IR queries express a combination of searches in different parts of (hierarchically) structured documents (see e.g., [38] for XML query examples). In the case of shallow hierarchical documents, such as the ones in TREC and CLEF collections, structured queries can be formed by adding one or more field-like structural constraints to the unstructured query. Looking at the description of the ad-hoc topic 305 in the TREC collection and knowing what are the most frequent terms within "subject" elements in the collection, we can come up with the following structured query, where the first part is actually the topic title (see Section 4.3 for more details on forming structured queries):

```
most dangerous vehicles
     SUBJECT: safety automobile accidents
```

Such queries provide the base for exploring the usefulness of structured information in poorly structured documents, such as TREC and CLEF data collections, and for finding what is the best way of incorporating this additional information with the traditional document retrieval. The results on structural search are presented in Section 5.2.

3. RETRIEVAL SYSTEM AND MODELS

In this section we describe our three-level database system, called TIJAH, and explain the instantiation of different retrieval models in score region algebra.

3.1 Retrieval system architecture

The TIJAH system consists of a conceptual, logical, and physical level. The central part of the system is the transparent logical level based on *score region algebra* (*SRA*) [24]. The transparency is reflected by the ability to instantiate different retrieval models without affecting the specification of the logical operators, while keeping the same query language at the conceptual level and with the straightforward physical implementation using arbitrary (low-level) DBMS.

3.1.1 Conceptual Level

Since TIJAH was originally developed for XML retrieval, on conceptual level we use the syntax of the Narrowed Extended XPath I (NEXI) query language [38]. NEXI allows only descendant steps from XPath 1.0 [8] and introduces an *about* clause that specifies which part of the document should be searched and what are the query terms. Therefore, the simple (*title*) TREC ad-hoc query 305 can be expressed in NEXI as:

```
//DOC[about(., most dangerous vehicles)]
```

The query is first processed at the conceptual level where stemming and stopword removal is performed. Additionally, the retrieval model and its parameters are specified. The conceptual query, together with the retrieval model specification, is then forwarded to the logical level.

3.1.2 Logical level

The logical level is based on the score region algebra [24, 26] that regards a document as a set of regions, where each region represents an XML (or SGML) element or a term in a document (for more details on region modeling see [25]). A region is defined by its starting position (s), end position

Table 1: Score region algebra operators.

Operator	Operator definition
$\sigma_{n=name,t=type}(R)$	$\{r \mid r \in R \wedge r.n = name \wedge r.t = type\}$
$\nabla(R)$	$\{(r.s, r.e, r.n, r.t, f_{prior}(r)) \mid r \in R\}$
$R_1 \sqsupset_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\sqsupset}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$
$R_1 \sqcap_p R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, r_1.p \otimes r_2.p) \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1, e_1, n_1, t_1) = (s_2, e_2, n_2, t_2)\}$
$R_1 \sqcup_p R_2$	$\{(r.s, r.e, r.n, r.t, r_1.p \oplus r_2.p) \mid r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((r.s, r.e, r.n, r.t) = (r_1.s, r_1.e, r_1.n, r_1.t) \vee (r.s, r.e, r.n, r.t) = (r_2.s, r_2.e, r_2.n, r_2.t))\}$
$R_1 \blacktriangleright R_2$	$\{(r_1.s, r_1.e, r_1.n, r_1.t, f_{\blacktriangleright}(r_1, R_2)) \mid r_1 \in R_1 \wedge r_1.t = node\}$

(e), type (t) that can be either *term* or *node*, name (n), and score (p). The score information depicts the score of a region with respect to the query.

SRA is organized along four elementary retrieval aspects, as we mentioned before: element/term selection, element relevance score computation, relevance score combination, and relevance score propagation. The set of operators we use is depicted in Table 1. The first operator models the aspect of element selection and selects either elements or terms in a document. The operator $\nabla(R)$ models the element prior and is used in combination with models that need priors to improve effectiveness.

Operator $\sqsupset_p$ models the element score computation aspect based on the abstract function $f_{\sqsupset}(r_1, R_2)$. The abstract function implements the basic retrieval model that assigns a score to a region r_1 looking at its starting position, end position, and score value. It also takes into account scores of contained regions from the region set R_2. For example, it can be instantiated using term frequency and document frequency for the tf.idf model.

Operators $\sqcap_p$ and $\sqcup_p$ model the score combination aspect in an AND or OR logical combination of sets of regions, transparently implemented using abstract operators $\otimes$ and $\oplus$ respectively. The last operator in Table 1 models upwards score propagation and is used for queries that include document structure. The instantiation of upwards score propagation function $f_{\blacktriangleright}(r_1, R_2)$, as well as other transparent functions and operators, is presented in the next section.

Using the operators given in Table 1 we can easily express the conceptual query in SRA at the logical level. For example, query 305 can be expressed as:

$$(\sigma_{t=node,n=\text{'DOC'}} \sqsupset_p \sigma_{t=term,n=\text{'most'}})$$
$$\sqcap_p (\sigma_{t=node,n=\text{'DOC'}} \sqsupset_p \sigma_{t=term,n=\text{'dangerous'}})$$
$$\sqcap_p (\sigma_{t=node,n=\text{'DOC'}} \sqsupset_p \sigma_{t=term,n=\text{'vehicles'}})$$

Such a query plan, with the right retrieval model specification, is then passed to the physical level for execution.

3.1.3 Physical level

The physical implementation in the TIJAH system is done using MonetDB kernel [3]. MonetDB is a low level database engine. For the communication with the kernel we use the Monet Interpreter Language (MIL). Therefore, each operator at the logical level is implemented as one MIL function, or as many in case of transparent operators for score computation, combination, and propagation. Based on the logical query plan and retrieval model specification, the call to the proper MIL functions with specified parameters is achieved. The output of the query execution, which is a list of elements with their respective scores reflecting the relevance of an element/document to a query, is then sorted and transformed into the proper output for evaluation.

3.2 Instantiating the retrieval model

In TIJAH (SRA) retrieval models are instantiated along the four elementary retrieval aspects. They are defined at the logical level but their real implementation is at the physical level. Element and term selection operator always selects elements with the computed or default score (in case R is the set of all regions in the collection), as can be seen in Table 1. For the default score for all regions in the collection we choose 1.0. We use only one specification for the element/term selection aspect: term regions are always selected with the usage of stemming and element regions are selected by strictly matching their names. For the other three retrieval aspects we use score region algebra transparency, and several definitions for functions representing each aspect are explained below.

For transparently specifying scoring functions we use auxiliary functions. In these functions $r_i \prec r_j$ is used to denote that the region r_i is contained in the region r_j ($r_i \prec r_j \Leftrightarrow s_j < s_i \leq e_i < e_j$) and C is used to denote the set of all regions in the collection. The auxiliary functions are used for a number of purposes. The first one counts the number of regions in the region set R, denoted with $|R|$. The second one computes the size of the region r, either based on starting and end index of the region (Equation 1) or the number of contained terms (Equation 2). Finally, the third one computes the average size of the regions with the region name n in the collection, denoted with $avg\_size(n)$ (Equation 3), also based either on the element index or term count.

$$size^{element}(r) = r.e - r.s - 1 \tag{1}$$

$$size^{term}(r) = |\{r_1 \in C | r_1.t = term \wedge r_1 \prec r\}| \tag{2}$$

$$avg\_size(n) = \frac{\sum_{r_1 \in C | r_1.n = n} size(r_1)}{|\{r_1 \in C | r_1.n = n\}|} \tag{3}$$

The difference between Equation 1 and Equation 2 is that the former, beside terms, also counts the opening and closing tags of regions contained in the region r.

For computing the prior we used a length prior as given in Equation 4. The length prior is based on the assumption that the larger elements are more likely to be the right answers to a query.

$$f_{prior}(r) := r.p \cdot size(r) \tag{4}$$

3.2.1 Element (relevance) score computation

Operator $\sqsupset_p$ models element relevance score computation, i.e., the concept that the search elements (regions in the first operand) should contain the term (regions in the second operand). Therefore, the function $f_{\sqsupset}(r_1, R_2)$, applied to a region r_1 and a 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. We use five retrieval models to specify score computation, two baseline ones: Boolean and tf.idf [34], and three more advanced: Language Models (LMs) [18], the Okapi (INQUERY) model [6, 31], and the Garden Point XML (GPX) model [15][3].

The simple Boolean formula for score computation is given by:

$$f_{\sqsupset}^{\text{Bool}}(r_1, R_2) = r_1.p \cdot sgn(|\{r_2 \in R_2 | r_2 \prec r_1\}|) \quad (5)$$

For the tf.idf approach we use the following formula:

$$f_{\sqsupset}^{\text{tf.idf}}(r_1, R_2) = r_1.p \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot ln \frac{|\{r \in C | r.n = r_1.n\}|}{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}|} \quad (6)$$

The language model can be instantiated using auxiliary functions as:

$$f_{\sqsupset}^{\text{LMs}}(r_1, R_2) = r_1.p \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{size(r_1)} + (1-\lambda) \frac{|R_2|}{size(Root)}) \quad (7)$$

The Okapi formula is derived from the original model by removing the third factor from the product (see [31]). The reason is that the third factor is based on a size of the query and it is not supported in other models. The complex function $f_{\sqsupset}^{\text{Okapi}}$ is specified as:

$$f_{\sqsupset}^{\text{Okapi}}(r_1, R_2) = r_1.p \cdot ln \frac{|\{r \in C | r.n = r_1.n\}| - |\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}| + 0.5}{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}| + 0.5} \cdot \frac{(k_1+1) \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{k_1((1-b) + b \frac{size(r_1)}{avg\_size(r_1.n)}) + \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p} \quad (8)$$

The element relevance score computation in the GPX model is specified as:

$$f_{\sqsupset}^{\text{GPX}}(r_1, R_2) = r_1.p \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p}{|R_2|} \quad (9)$$

Although this is a slightly different formula than the original GPX model it showed equally good performance in our previous experiments [24].

To be able to apply different score combination and score propagation functions on all score computation models, we had to normalize the score computation function results for Okapi and tf.idf models to a range $[0, 1]$. The resulting scores of Boolean, GPX, and language model score computation functions are already normalized.

3.2.2 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 $\sqcup_p$. We make different choices for the implementation of abstract score combination operators.

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

Besides the simple implementations such as sum, product, minimum, and maximum ($\{\oplus, \otimes\} := \{+, *, min, max\}$), following [6] and [15], we also define these two abstract operators as probabilistic sum (shown in Equations 10) and exponential sum (shown in Equation 11):

$$r_1.p\{\oplus^{prob}, \otimes^{prob}\}r_2.p = 1 - (1 - r_1.p) \cdot (1 - r_2.p) \quad (10)$$

$$r_1.p\{\oplus^{exp}, \otimes^{exp}\}r_2.p = \begin{cases} r_1.p + r_2.p & \text{if } r_1.p = 0 \vee r_2.p = 0 \\ A \cdot (r_1.p + r_2.p) & \text{otherwise} \end{cases} \quad (11)$$

Although Equation 11 in combination with the formula in Equation 9 results in a retrieval model that is slightly different than the original GPX model, it does follow the semantics of the model which is to boost the scores for regions that contain more query terms.

3.2.3 Element score propagation

The operator $\blacktriangleright$ specifies propagation of scores to the containing elements, e.g., from SUBJECT to DOC elements in the TREC collection. Thus, the function $f_{\blacktriangleright}(r_1, R_2)$ specifies upwards element score propagation. It is instantiated either as a simple sum of scores (Equation 12) or as a weighted sum normalized by the size of regions in the left operand (Equation 13). We also perform smoothing for the score propagation to avoid zero scores for, e.g., documents that do not contain SUBJECT element. The score propagation smoothing uses the frequency of regions (elements) from the right operand contained in the regions (elements) having the name n_1 (name of the region in the left operand). The influence of this "element frequency" is regulated by a smoothing parameter ω.

$$f_{\blacktriangleright}^{\text{sum}}(r_1, R_2) = r_1.p \cdot (\omega \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p + (1-\omega) \cdot \frac{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}|}{|\{r \in C | r.n = r_1.n\}|}) \quad (12)$$

$$f_{\blacktriangleright}^{\text{wsum}}(r_1, R_2) = r_1.p \cdot (\omega \cdot \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} r_2.p \cdot size(r_2)}{size(r_1)} + (1-\omega) \cdot \frac{|\{r \in C | r.n = r_1.n \wedge \exists r_2 \in R_2 \wedge r_2 \prec r\}|}{|\{r \in C | r.n = r_1.n\}|}) \quad (13)$$

4. STRUCTURING QUERIES

In this section we present our experimental setup and explain structured query formulation of faceted and field-based structured queries.

4.1 Test collections

For the experimental evaluation we use four sub-collections from the TREC 6 collection: Foreign Broadcast Information Service (*fbis*), Federal Register (*fr94*), Financial Times (*ft*), and Los Angeles Times (*latimes*), as well as two Dutch newspapers from CLEF: Algemeen Dagblad (*ad*) and NRC Handelsblad (*nh*). The topic set consists of TREC topics 301 to 350, and three sets of CLEF topics: 41 to 90 91 to 140, and 141 to 200. Accordingly, we use the relevance assessments and *trec_eval* evaluation tool to test the effectiveness of distinct query specification and different retrieval models.

The unstructured queries that we use are title, title + description, and title + description + narrative queries from TREC and CLEF topics. We formed three types of structured queries: faceted queries, field-based structured queries, and the combination of the two. Additionally, we use the list-of-term variant of the faceted queries (called expanded queries), and compare it with other structured and unstructured queries.

4.2 Faceted queries

Faceted queries are designed based on the following approach. The user first analyses the request and identifies what are the most important keywords in his request and than groups these keywords into facets. As an example, for our TREC topic 305 given in Section 1 we can group the following terms:

```
vehicles crash cars crashworthy death danger
```

in two facets:

{(`vehicles, cars`),(`crash crashworthy death danger`)}.

The final step is combining terms from the same facet using the OR operator, and combining these faceted expressions using the AND operator. In such a way we obtain the NEXI query that looks like:

```
//DOC[(about(., vehicles) OR about(., cars)) AND
   (about(., crash) OR about(., crashworthy) OR
     about(., death) OR about(., danger))]
```

Thanks to Schiettecatte [35] we were in position to use faceted (Boolean) queries on the TREC collection. We use the whole set of 50 faceted queries for TREC. As we did not have such queries for the CLEF collection we developed them ourselves (see the following section) for CLEF 41-90 and CLEF 91-140 topic sets[4].

To test whether query expansion, without taking into account the classification of query terms and topic description and narrative fields, can improve the effectiveness of retrieval models, we perform several experiments on TREC and CLEF *expanded queries*. The queries are formed based on faceted queries, i.e., we only use the terms from the faceted query and remove the connectors (AND and OR) between terms. An example NEXI query expression for expanded TREC 305 topic is:

```
//DOC[about(., vehicles cars crash
                  crashworthy death danger)]
```

4.3 Field-based structured queries

To structure the queries we first analyzed the two selected document collections to see what kind of useful information we can extract and how we can use it to help the user in stating his query. In our experiments with field-based queries we only use the LA Times sub-collection of TREC 6 and the complete CLEF collection as they come with appropriate structure. The other parts of the TREC collection are not used as they either do not contain meaningful structural elements or it is too difficult to utilize information contained in the structural elements for the query formulation.

[4]Due to the space limitation, the queries used in our experiments in this paper can be found on the web: http://www.cs.utwente.nl/∼vojkan/TREC&CLEF-topics.

The LA Times collection contains three types of elements (tags) that can potentially be used for structured search: SUBJECT, SECTION, and TYPE. However, except for the SUBJECT part, the other two were not suitable for query formulation as the keywords present in these tags are not well classified and they are difficult to utilize by a user when forming a query. Additionally, there are many keywords occurring only several times and a few very frequent ones. On the other hand, the CLEF collection comes with a DTD and is far better organized. Among others, it also contains four informative elements that can be used for making structural queries: GEO (location), HTR (keywords), SEC (section), and PER (persons). Furthermore, unlike in the LA Times collection, tags are populated by a predefined, more thoughtfully chosen, set of terms.

A potential problem for experiments with field-based queries was that in both collections only a part of the documents contain these useful tags. For example, the TREC collection has a SUBJECT tag in 46 849 out of 131 896 documents. It is similar with GEO, HTR, and PER tags in the CLEF collection. Despite this, we decided to use it in our experiments, arguing that well organized collections, with the complete and consistent document annotation, would probably give even better results on structured queries.

For the TREC collection we formed structural queries in two steps. We first select the most frequently occurring (more than 500 times) keywords in the SUBJECT tags. Then, for each query, we choose the terms that are most relevant to the query and add them to the original or faceted query. For example, the original TREC query 305 with added structural constraints looks like:

```
//DOC[(about(., most dangerous vehicles) AND
  about(.//SUBJECT, safety automobile accidents)]
```

We added structural constraints to 41 out of 50 TREC queries. The results of the evaluation of structured TREC and CLEF queries are given in the next section.

To make structured queries out of CLEF topics we asked several of our Dutch colleagues to complete advanced search forms similar to the advanced search forms in for instance Google or PubMed. Our "users" would first read the CLEF topic description and narrative and fill in the following:

- what would be an exhaustive query that they will issue (using TREC faceted queries as an example)?
- what are the persons involved in the query?
- select from a drop-down box whether the topic is about domestic issues ("binenland") or foreign issues ("buitenland")?
- select from a drop-down box in which section of the newspaper the article is likely to be found?
- select from a drop-down box where is the location of the event (continent and country/city)?
- select from a drop-down box what are the keywords that could help the search?

For all the fields with drop-down boxes, an additional "no preference" option is provided. The options for section, locations, and keywords are automatically extracted from the collection. The "searcher's" input is then transformed into a faceted query, as stated in the previous section, and a field-based query. For example, for CLEF topic 60 the field-based

Table 2: Query types used in the experiments illustrated on NEXI queries for TREC topic 305.

Query type	Abbr.	NEXI query
title	**T**	`//DOC[about(., most dangerous vehicles)]`
expanded	**E**	`//DOC[about(., vehicles cars crash crashworthy death danger)]`
title + description	**TD**	`//DOC[about(., most dangerous vehicles which are the most crashworthy and least crashworthy passenger vehicles)]`
title + description + narrative	**TDN**	`//DOC[about(., most dangerous vehicles which are the most crashworthy and least crashworthy passenger vehicles a relevant document will contain information on the crashworthiness ... miles for example)]`
faceted	**F**	`//DOC[(about(., vehicles) OR about(., cars)) AND (about(., crash) OR about(., crashworthy) OR about(., death) OR about(., danger))]`
field-based + title	**ST**	`//DOC[(about(., most dangerous vehicles) AND about(.//SUBJECT, safety automobile accidents)]`
field-based + faceted	**SF**	`//DOC[(about(., vehicles) OR about(., cars)) AND (about(., crash) OR about(., crashworthy) OR about(., death) OR about(., danger)) AND about(.//SUBJECT, safety automobile accidents)]`

structural query expressed in NEXI looks like:

```
//DOC[about(., de franse corruptieschandalen)
    AND about(.//HTR, fraude en corruptie)
    AND about(.//SEC, buitenland)
    AND about(.//GEO, europa frankrijk)]
```

5. EXPERIMENTS

In this section we report the experimental results on the CLEF and TREC collections using unstructured and structured queries, as well as using different implementations for score computation, combination, and propagation aspects. The mean average precision (MAP) values are reported for all the experiements. We first analyze the results of our preliminary experiments on unstructured queries, and then discuss the usage of structure in document retrieval. The complete list of query types, along with the shorthand notation and example NEXI queries formed out of TREC topic 305, is presented in Table 2.

5.1 Unstructured queries

Below we discuss experimental results obtained using unstructured queries.

5.1.1 Best parameters and size estimation

In our preliminary set of experiments on the CLEF 141-200 topics we try to find the best way to compute the size of the regions and appropriate parameters for score computation and score combination aspects, instantiated for different retrieval models[5]. The queries that were used to determine optimal parameter settings were not used in the experiments in the following sections. Although we assumed that element size computation based on region indexes (Equation 1) would be less effective than term count (Equation 2), especially because most of the retrieval systems so far use the latter, this is shown to be a false assumption. This fact is important since precomputations are not used in TIJAH. As a consequence, element size computation is approximately twice as fast as the term count.

Parameter λ in Equation 7 is estimated to be 0.4 which is also the value that is frequently used for language models in structured retrieval [26]. The best values of parameters for the Okapi score computation function (Equation 8) are $k_1 = 0.7$ and $b = 0.4$. These are relatively low values with respect to the usual values for k_1 (1.2) and b (0.75), but this might be caused by excluding the third factor in the product from the Okapi model. To estimate the value of parameter A in the score combination function, used to model the GPX system (Equations 9 and 11), we ran a set of experiments, where $10 > A > 0$, and obtained a relatively stable MAP for $A > 4$. For our further experiments we chose to use $A = 7$.

[5]As the results of this series of experiments are not of a primary interest for the paper we do not illustrate them.

5.1.2 Experiments with unstructured queries

The results of our experiments using title (**T**), expanded (**E**), title + description (**TD**), and title + description + narrative (**TDN**) queries are depicted in Table 3. In these series of experiments we explore what is the best AND score combination for different score computation implementation and then analyze the impact of distinct query formulations on retrieval effectiveness.

We test five different score combination functions: sum, product, minimum, maximum, and probabilistic sum (Equation 10), on Boolean, GPX, LMs, Okapi, and tf.idf score computation functions, and exponential sum (Equation 11) on GPX. By analyzing the two best performing AND score combination implementations for each score computation function, depicted in Table 3, we can see that advanced retrieval models by far outperform the baseline ones (Boolean and tf.idf). Furthermore, for most of the advanced models the score combination specified in the original formula of these retrieval models is actually the best AND score combination function. The only exceptions are the tf.idf model where the sum is not in the two best performing combination functions, and the Okapi model where the probabilistic implementation (Equation 10) shows comparable performance to the score combination implemented as sum.

Looking at different query formulations, the effectiveness is much higher for Boolean and tf.idf models on title and expanded queries than on long **TD** and **TDN** queries. This is expected due to the score computation specification for Boolean and tf.idf models (Equations 5 and 6). However, this is not the case for the advanced models where in most of the cases the mean average precision values are comparable among **T**, **E**, **TD**, and **TDN** queries. The MAP values given in bold represent the best run for each score computation model on one topic set.

By comparing original title queries with the expanded ones we can conclude that in many cases the simple title queries outperform the expanded ones (especially for the TREC experiments). In all cases **TD** queries outperform longer **TDN** queries, but it is not clear whether **TD** or ti-

Table 3: Unstructured queries: experimental results for the two best-performing AND score combination functions on title (T), expanded (E), title + description (TD), and title + description + narrative (TDN) queries.

	AND	TREC				CLEF 41 − 90				CLEF 91 − 140			
Model	(⊗)	T	E	TD	TDN	T	E	TD	TDN	T	E	TD	TDN
Bool	min	0.0803	0.0317	0.0018	0.0000	0.1288	0.0681	0.0102	0.0000	0.1223	0.0051	0.0081	0.0002
Bool	prod.	0.0803	0.0317	0.0018	0.0000	0.1288	0.0681	0.0102	0.0000	0.1223	0.0051	0.0081	0.0002
tfidf	min	0.1004	0.0329	0.0018	0.0000	0.1541	0.0701	0.0122	0.0000	0.1600	0.0051	0.0081	0.0002
tfidf	prod.	0.1185	0.0493	0.0019	0.0000	0.1698	0.0788	0.0133	0.0000	0.1800	0.0054	0.0095	0.0002
GPX	exp.	**0.1997**	0.1960	0.1308	0.0143	0.2752	**0.3137**	0.2535	0.1246	**0.2896**	0.2613	0.2813	0.1791
GPX	prod.	0.1514	0.0886	0.0734	0.0559	0.2503	0.1904	0.1776	0.0782	0.2370	0.0714	0.1127	0.0962
LMs	min	0.1265	0.0467	0.0339	0.0053	0.1900	0.1025	0.0598	0.0030	0.2206	0.0374	0.0487	0.0133
LMs	prod.	0.2223	0.2205	**0.2230**	0.1878	0.3080	**0.3661**	0.3594	0.3302	0.3211	0.3370	**0.3866**	0.3658
Okapi	prob.	0.2205	0.2232	**0.2239**	0.1587	0.3246	**0.3728**	0.3575	0.3240	0.3237	0.3499	**0.3944**	0.3923
Okapi	sum	0.2205	0.2201	0.2184	0.1551	0.3257	0.3724	0.3592	0.3215	0.3247	0.3425	0.3900	0.3889

Table 4: Faceted queries (F): experimental results with different OR score combination functions.

Model	AND (⊗)	OR (⊕)	TREC	CLEF 41–90	CLEF 91–140
Bool	min	max	0.0759	0.1608	0.0665
Bool	min	prob.	0.0759	0.1608	0.0665
tfidf	prod.	prob.	0.1166	0.2220	0.0965
tfidf	prod.	sum	0.1167	0.2220	0.1034
GPX	exp.	exp.	0.1849	0.2965	0.2344
GPX	exp.	min	0.1603	0.2691	0.2413
LMs	prod.	prob.	**0.2582**	0.3751	0.3450
LMs	prod.	sum	0.2580	**0.3754**	0.3450
Okapi	prob.	max	0.2499	0.3857	0.3773
Okapi	prob.	prob.	0.2207	0.3633	0.3217
Okapi	sum	max	**0.2559**	**0.3886**	0.3765
Okapi	sum	prob.	0.2329	0.3711	0.3317

tle/expanded queries give better results. While on *TREC* topics there is almost no difference in the results (except for GPX model), on *CLEF 41–90* expanded queries outperform **TD** and **TDN** queries, and on *CLEF 91–140* **TD** and **TDN** queries outperform **T** and **E** queries. This indicates that throwing relevant terms in the query, without properly classifying them, does not necessarily leads to higher effectiveness. The question still remains if the faceted query specification can outperform both title and expanded queries, as well as **TD** and **TDN** queries.

5.2 Structured queries

The analysis of the retrieval effectiveness on structured queries is presented below.

5.2.1 Faceted queries

The goal of this experiment series is to test how systems can benefit from faceted query formulation and to find what is the best OR score combination instantiation for the best AND score combination function determined in the previous section. The results for the two best compositions of AND and OR score combination functions for Boolean, GPX, LMs, and tfidf score computation functions, and four in case of the Okapi score computation function, are depicted in Table 4. We report four compositions for the Okapi score computation function as it gives high MAP values in combination with the AND score combination implemented as probabilistic sum or sum (see Table 3).

If we compare the results from the expanded queries in Table 3 and the results from the faceted queries in Table 4, we can see no consistent improvements in the MAP values for the Boolean, GPX, and tf.idf models, except in the *CLEF 41–90* experiments. However, for the language models and Okapi, for both collections and all three topic sets, there is a consistent improvement over both title and expanded queries. Furthermore, in most of the cases and for each score computation function, AND and OR score combination functions exist for which the faceted run outperform the extended one. This shows that *structuring queries can help improving effectiveness.*

By comparing the MAP values for faceted runs with the MAP values for the **TD** and **TDN** runs on advanced queries we can see that on *TREC* and *CLEF 41–90* topics faceted queries give better results than **TD** and **TDN** queries (given in bold). For example, for the TREC data, the MAP increases up to 16% for Okapi with AND combination implemented as sum and OR combination as maximum, with respect to **TD** and **TDN** (as well as **T** and **E**) queries. On *CLEF 91–140* this is not the case, but it can be due to the high complexity of manually generated faceted queries. However, *faceted queries showed at least comparable performance to long **TD** and **TDN** queries.*

Table 4 also shows that more than one OR score combination function that has a high MAP exists: LMs with sum and probabilistic sum implementation and both Okapi models (AND combination implemented as sum or probabilistic sum) with maximum or probabilistic sum as OR combination.

5.2.2 Field-based structured queries

In the last set of experiments we explore how structured information can be incorporated in retrieval models for document retrieval. Then we test whether we can further improve the precision-recall values by adding field-based structural constraints to the query and using the information that is tagged in a document. The structural constraints are added to the original (*title*) and *faceted* queries. We report three advanced models (GPX, language models, and Okapi) with the best AND and OR score combination functions and using two score propagation functions. The functions given in Equations 12 and 13 (sum and $wsum$) are employed to test whether and up to what degree we can improve the effectiveness obtained with unstructured queries.

With the first set of experiments we try to find out how structured information should be combined with the "document search". In other words, what are the values of the element smoothing parameter ω that give the highest MAP for different retrieval models. Furthermore we want to find out which of the two implementations of the score propagation function are more effective.

Table 5: Field-based structured queries formed using topic title (ST): estimating the best value of smoothing parameter ω (from 0.1 to 0.95) on LA Times collection.

Model, $\otimes$, $\oplus$	$f_{\blacktriangleright}$	T	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95
GPX, exp., exp.	sum	0.2527	0.2530	0.2530	0.2529	0.2530	**0.2531**	0.2517	0.2518	0.2518	0.2518	0.2519
GPX, exp., exp.	wsum	0.2527	0.2529	0.2529	0.2529	0.2529	0.2529	0.2529	0.2529	0.2529	0.2529	0.2529
LMs, prod., sum	sum	0.2804	0.2807	0.2808	0.2810	0.2810	0.2814	0.2818	0.2827	0.2828	**0.2863**	0.2860
LMs, prod., sum	wsum	0.2804	0.2804	0.2805	0.2805	0.2805	0.2805	0.2805	0.2806	0.2806	0.2810	**0.2812**
Okapi, sum, max	sum	0.2819	0.2822	0.2844	0.2868	0.2877	**0.2882**	0.2859	0.2849	0.2822	0.2784	0.2766
Okapi, sum, max	wsum	0.2819	0.2820	0.2837	0.2839	**0.2840**	0.2838	0.2838	0.2837	0.2839	0.2836	0.2836

Table 6: Comparison of field-based structured queries formed using title (ST) and faceted (SF) queries, and title (T) and faceted (F) queries: experimental results on CLEF collection.

Model, $\otimes$, $\oplus$	propagation ($f_{\blacktriangleright}$)	CLEF 41 − 90				CLEF 91 − 140			
		T	ST	F	SF	T	ST	F	SF
GPX, exp., exp.	sum, $\omega = 0.5$	0.2752	0.2872	0.2965	0.3071	0.2896	0.2965	0.2344	0.2350
LMs, prod., sum	sum, $\omega = 0.95$	0.3080	0.3314	0.3754	0.3853	0.3211	0.3161	0.3450	0.3468
Okapi, sum, max	sum, $\omega = 0.5$	0.3257	0.3760	0.3886	0.4313	0.3247	0.3443	0.3765	0.3908

We use the LA Times collection for estimating ω. The results are depicted in Table 5. Looking at the results, there is no significant difference in using different values of ω. However, for models that use sum-like AND score combination (sum and exponential sum) $\omega \sim 0.5$ gives slightly better results, while high values of ω are better for the AND score combination implemented as product. The best MAP values for each model and score propagation function are given in bold. Furthermore, in all cases score propagation implemented as *sum* gives higher MAP values than score propagation implemented as weighted sum (*wsum*). Therefore, or the following experiments we use the Equation 12 and $\omega = 0.5$ for GPX and Okapi based models and $\omega = 0.95$ for language model variants.

The mean average precision values of our structured experiments on CLEF collection are reported in Table 6. The results clearly indicate that the structural constrains added to the query improve the mean average precision for different variations of language models and Okapi (in 11 out of 12 runs presented in Table 6) with respect to the title and expanded queries. Also, the effectiveness of structured queries is far better on *CLEF 41–90* than effectiveness of **TD** and **TDN** queries, while on *CLEF 91–140* the results are comparable. We can also see that the Okapi model with score combination modeled as sum ($\otimes$) and max ($\oplus$) is more robust to score propagation and more effective than LM in all of its variants.

6. CONCLUSIONS AND FUTURE WORK

In this paper we showed that the score region algebra is a useful framework for analyzing document retrieval with respect to unstructured as well as structured queries. We investigated the composition of different implementations of structured retrieval aspects, i.e., computation, combination, and propagation of scores, for complex structured query formulations. Furthermore, we illustrate how state of the art retrieval models, such as language models and Okapi, can benefit from the formulation of structured queries, having such a flexible structured retrieval framework.

We evaluated unstructured (title and expanded) and structured (faceted and field-based) queries on TREC and CLEF test collections. The effectiveness is improved when using faceted queries, queries that utilize document structure, as well as the combination of faceted and structured queries. Furthermore, the effectiveness of structured queries is comparable or better than when using long title + description or title + description + narrative queries.

We have also shown that the score computation based on Okapi and language models work well with several score combination functions. For example, Okapi shows good results for AND score combination implemented as probabilistic sum or sum and OR score combination implemented as maximum, while language model shows good results with AND score combination implemented as product and OR score combination implemented as probabilistic sum. The experiments illustrate also that in structured retrieval summing the scores of containing elements when propagating them to the document (element) gives high effectiveness.

In the future we want to extend our research to semantically richer collections, and continue our experimentation on highly structured documents, such as XML (e.g., [14]). Our aim is to improve the models used for structured document retrieval. Additionally we want to compare the outcome of the experiments presented in this paper to retrieval on documents that have more complex structure.

7. REFERENCES

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th WWW Conference*, 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto, editors. *Modern Information Retrieval*. Addison Wesley Longman, 1999.

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

[4] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic Query Expansion Using SMART: TREC 3. In *Text REtrieval Conference (TREC)*, 1994.

[5] F.J. Burkowski. Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Texts. In *Proceedings of the 15th ACM SIGIR Conference*, 1992.

[6] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the 3rd DEXA Conference*, 1992.

[7] James P. Callan, W. Bruce Croft, and John Broglio. TREC and TIPSTER Experiments with INQUERY.

Information Processing and Management: an International Journal, 31(3), 1995.

[8] J. Clark and S. DeRose. XML Path Language Version 1.0, W3C, http://www.w3.org/tr/xpath.

[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), 1995.

[10] W.B. Croft, R. Cook, and D. Wilder. Providing Government Information on the Internet: Experiences with THOMAS. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries (DL)*, pages 19–24, 1995.

[11] W.B. Croft and H. Turtle. A Retrieval Model for Incorporating Hypertext Links. In *Proceedings of the Second Annual ACM Conference on Hypertext (HYPERTEXT)*, pages 213–224. ACM Press, 1989.

[12] E.N. Efthimiadis. Query Expansion. *Annual Review of Information Systems and Technology*, 31(4):121–187, 1996.

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

[14] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the Third Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, 2005.

[15] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[16] D. Hawking, N. Craswell, F. Crimmins, and T. Upstill. How Valuable is External Link Evidence when Searching Enterprise Webs? In *Proceedings of the 15th Australasian Database Conference (ADC)*, pages 77–84. Australian Computer Society, Inc., 2004.

[17] M. Hearst. *User Interfaces and Visualization*, chapter 5. In Baeza-Yates and Ribeiro-Neto [2], 1999.

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

[19] B. J. Jansen. The Effect of Query Complexity on Web Searching Results. *Information Research*, 6(1), 2000.

[20] J. Kaamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Understanding Content-and-Structure. In *Proceedings of the INEX Workshop on Element Retrieval Methodology*, 2005.

[21] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Structured Queries in XML Retrieval. In *Proceedings of the ACM CIKM Conference*, pages 4–11, 2005.

[22] D. Metzler and W.B. Croft. Combining the Language Model and Inference Network Approaches to Retrieval. *Information Procesing and Management*, 40(5):735–750, 2004.

[23] A. Micarelli, F. Gasparetti, F. Sciarrone, and S. Gauch. Personalized Search on the World Wide Web. In *The Adaptive Web: Methods and Strategies of Web Personalization*, 2005.

[24] 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 CIKM Conference*, pages 12–19, 2005.

[25] 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)*, 2004.

[26] V. Mihajlović, G. Ramírez, A. P. de Vries, D. Hiemstra, and H. E. Blok. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In *Proceedings of the 3rd INEX Workshop, LNCS 3493, Springer*, 2005.

[27] G.A. Miller, C. Fellbaum, R. Tengi, S. Wolff, P. Wakefield, H. Langone, and B. Haskell. WordNet: A Lexical Database for the English Language.

[28] P. Ogilvie and J. Callan. Using Language Models for Flat Text Queries in XML Retrieval. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[29] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. Terrier Information Retrieval Platform. In *Proceeding of the 27th European Conference on Information Retrieval Research (ECIR)*, pages 517–519. Lecture Notes in Computer Science, Springer, March 2005.

[30] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized Search. *Communications of the ACM*, 45(9):50–55, 2002.

[31] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th ACM SIGIR Conference*, 1994.

[32] A. Salminen and F.W. Tompa. PAT Expressions: An Algebra for Text Search. In *Proceedings of COMPLEX*, 1992.

[33] G. Salton and C. Buckley. Improving Retrieval Performance By Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[34] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGrow-Hill, 1st edition, 1983.

[35] F. Schiettecatte. Document Retrieval Using the MPS Information Server (a report on the TREC-6 experiment). In *Proceedings of the Sixt Text Retrieval Conference (TREC)*, pages 477–488, 1998.

[36] U. Shah, T. Finin, and A. Joshi. Information Retrieval on the Semantic Web. In *Proceedings of the eleventh International Conference on Information and Knowledge Management (CIKM)*, pages 461–468. ACM Press, 2002.

[37] T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. Indri: A Language Based Search Engine for Complex Queries. In *Proceedings of the International Conference on Intelligence Analysis*, 2005.

[38] A. Trotman and R. A. O'Keefe. The Simplest Query Language That Could Possibly Work. In *Proceedings of the 2nd INEX Workshop*, ERCIM Publications, 2004.

[39] J. Xu and W.B. Croft. Query Expansion Using Local and Global Document Analysis. In *Proceedins of the 19th SIGIR Conference*, pages 4–11, 1996.