TIJAH at INEX 2004
Modeling Phrases and Relevance Feedback

Vojkan Mihajlović[1], Georgina Ramírez[2], Arjen P. de Vries[2], Djoerd Hiemstra[1], and Henk Ernst Blok[1]

[1] University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands
{v.mihajlovic, d.hiemstra, h.e.blok}@utwente.nl
[2] Centre for Mathematics and Computer Science,
P.O. Box 94079, 1090GB Amsterdam, The Netherlands
{georgina, arjen}@cwi.nl

Abstract. This paper discusses our participation in INEX using the TIJAH XML-IR system. We have enriched the TIJAH system, which follows a standard layered database architecture, with several new features. An extensible conceptual level processing unit has been added to the system. The algebra on the logical level and the implementation on the physical level have been extended to support phrase search and *structural* relevance feedback. The conceptual processing unit is capable of rewriting NEXI content-only and content-and-structure queries into the internal form, based on the retrieval model parameter specification, that is either predefined or based on relevance feedback. Relevance feedback parameters are produced based on the data fusion of result element score values and sizes, and relevance assessments. The introduction of new operators supporting phrase search in score region algebra on the logical level is discussed in the paper, as well as their implementation on the physical level using the pre-post numbering scheme. The framework for structural relevance feedback is also explained in the paper. We conclude with a preliminary analysis of the system performance based on INEX 2004 runs.

1 Introduction

In our research for INEX 2004 we extended the TIJAH system to support more advanced IR techniques, namely phrase search and relevance feedback. The TIJAH system follows a layered database architecture, consisting of a conceptual, a logical, and a physical level. Each level has been built upon a different data model and has its own operators. The top level is based on the NEXI query language [10]. A NEXI query is first translated (at the conceptual level) into an internal query representation that closely resembles the NEXI query language, but enriched with some additional operators. The translation process is based on the retrieval model specification. The conceptual query is then transformed into a score region algebra (SRA) query plan [7] on the logical level of the TIJAH system. SRA views XML as a collection of regions and not as a tree-like structure, and operators in the SRA are based on the region containment relation and on region frequency counts to support vague containment conditions. The logical query plan is transformed into the physical plan (via Monet interpreter language - MIL) that is executed in the MonetDB database kernel [1].

The TIJAH system that we use for INEX 2004 is an extended version of the TIJAH system used in 2003 [6]. Each level of the TIJAH database system has been extended to support phrase search and relevance feedback. Thus, the conceptual level is capable of handling phrases and supports relevance feedback specification. New operators have been introduced into the score region algebra to support phrase modeling and relevance feedback specification and the physical level has been enriched with new functions that implement phrase search. Furthermore, a fully automatic query rewriting unit has been developed at the conceptual level capable of transforming original NEXI queries into proper conceptual queries based either on the retrieval model specification or on the relevance feedback data.

The retrieval model used for the NEXI *about* function is essentially the same as the one used for INEX 2003 [6]. We calculate the relevance of a document component (i.e., XML element), following the idea of independence between the relevance on exhaustivity and the relevance on specificity. The relevance on exhaustivity is estimated using the language modeling approach to information retrieval [4]. The phrase model is kept orthogonal to the unigram language model for single terms, similarly to [9], and we used variants of the n-gram ($n > 1$) language model to see if and in what degree phrases can contribute to the TIJAH system effectiveness. The relevance on specificity is assumed to be related to the component length (e.g., following a log-normal distribution).

This paper presents our approaches for two out of five tracks defined for INEX 2004, namely the ad-hoc track and the relevance feedback track. For the ad-hoc track, we developed approaches for both the content-only (CO) and the vague content-and-structure (VCAS) subtasks. Different models have been implemented in the TIJAH system for these subtasks. Moreover, the TIJAH system supports the specification of relevance feedback parameters and a simple model for relevance feedback on structure has been implemented in our system.

The following section gives a global overview of the TIJAH system architecture. Section 3 describes the capabilities of a conceptual level of our system performing different NEXI query rewriting and expansions. Section 4 specifies an extension of score region algebra for phrase handling and explains how these expressions are mapped into efficient operations on the physical level. Section 5 describes the incorporation of relevance feedback on structure in our system. The paper concludes with a discussion of the experiments performed with the TIJAH system for the two INEX ad-hoc search tasks (CO and CAS) and for the INEX relevance-feedback task.

2 TIJAH System Architecture

The TIJAH XML-IR system follows a traditional three-level database architecture consisting of a conceptual, logical, and physical level. Although the concept has been well known in the database field for about thirty years, we introduced some modifications in the architecture to bridge the gap between traditional DBMSs and IR systems.

2.1 Conceptual level

As a base for the conceptual level we used the Narrowed Extended XPath (NEXI) query language [10] as proposed by the INEX community in 2003. The NEXI query language supports only a subset of the XPath syntax and extends XPath with a special *about* function that ranks XML elements by their estimated relevance to a textual query. As such, the invocation of the *about* function can be regarded as the instantiation of a retrieval model.

Throughout the paper we will use two NEXI examples, one taken from the INEX CAS topic 149:

```
//article[about(.//(abs|kwd), "genetic algorithm")]
              //bdy//sec[about(., simulated annealing)]
```

and the other from INEX CO topic 166:

```
+"tree edit distance" +XML -image
```

During query processing a (conceptual) NEXI query language expression is encoded into an internal representation that closely resembles the original query in its structure, and all manipulations are done on this internal representation. As a result of the processing on the conceptual level we obtain a conceptual query representation.

2.2 Logical level

The difference on the logical level of traditional DBMSs and our system is in that we enhanced it with an algebra that takes into account the specific (i.e., nested) structure of the modeled data, i.e., XML in our case, to enable high level reasoning about the query specification and algebraic optimization. Since the algebra supports region score manipulation and ranked retrieval we named it score region algebra (SRA).

The basic score region algebra operators that involve score manipulations are depicted in Table 1[3]. We assume that the default value for score attribute is 1. Note that the probabilistic containment operators $\sqsupset_p$, $\not\sqsupset_p$, $\blacktriangleright$, and $\blacktriangleleft$ copy the region start, end, type, and name attribute values from the left operand region set (R_1) to the result region set, while the score attribute of the result

[3] We used a slightly different notation than in [6]. For more extensive coverage of our score region algebra we refer to [7].

Table 1. Region algebra operators for score manipulation.

Operator	**Operator definition**
$\sigma_{t=type,n=name}(R)$	$\{r \vert r \in R \wedge t = type \wedge n = name\}$
$R_1 \sqsupset_p R_2$	$\{r \vert r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \times f_{\sqsupset}(r_1,R_2)\}$
$R_1 \not\sqsupset_p R_2$	$\{r \vert r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \times f_{\not\sqsupset}(r_1,R_2)\}$
$R_1 \blacktriangleright R_2$	$\{r \vert r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \times f_{\blacktriangleright}(r_1,R_2)\}$
$R_1 \blacktriangleleft R_2$	$\{r \vert r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \times f_{\blacktriangleleft}(r_1,R_2)\}$
$R_1 \sqcap_p R_2$	$\{r \vert r_1 \in R_1 \wedge r_2 \in R_2 \wedge (s_1,e_1,n_1,t_1) = (s_2,e_2,n_2,t_2)$ $\wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \otimes p_2\}$
$R_1 \sqcup_p R_2$	$\{r \vert r_1 \in R_1 \wedge r_2 \in R_2 \wedge ((s,e,n,t) := (s_1,e_1,n_1,t_1)$ $\vee (s,e,n,t) := (s_2,e_2,n_2,t_2)) \wedge p := p_1 \oplus p_2\}$

set (p) gets its value based on the containment relation among regions in the left and regions in the right operand as well as their respective score values. The definitions of the set-like operators ($\sqcap_p$ and $\sqcup_p$) are similar to the definitions of basic set intersection and set union operators, i.e., the result region start, end, type and name are obtained the same way as for set intersection and union operators, except that the result score value for regions is defined based on the score values of regions in the left and right operand region set.

In the definition of score operators we introduced four abstract scoring functions: $f_{\sqsupset}$, $f_{\not\sqsupset}$, $f_{\blacktriangleright}$, and $f_{\blacktriangleleft}$, as well as two abstract operators: $\otimes$ and $\oplus$, that define the retrieval model. For the $\oplus$ operator we assume that there exists a default value for the score (denoted with d), and in case the region r_1 is not present in the region set R_2 the score is computed as $p = p1 \oplus d$ and in case the region r_2 is not present in the region set R_1 the score is computed as $p = d \oplus p_2$.

The functions $f_{\sqsupset}$, $f_{\not\sqsupset}$, $f_{\blacktriangleright}$, and $f_{\blacktriangleleft}$, applied to a region r_1 and a region set R_2, result in the numeric value that takes into account the score values of all regions r_2 ($\in R_2$) and the numeric value that reflects the structural relation between the region r_1 and the region set R_2. The abstract $\otimes$ operator specifies how scores are combined in an `and` expression, while the $\oplus$ operator defines the score combination in an `or` expression inside the NEXI predicate. The exact instantiation of these functions and operators is done on the physical level as can be seen in the next section.

2.3 Physical level

The SRA is defined as an XML specific logical algebra and can be implemented easily with relational operators [6, 11, 2]. Since we used the MonetDB on the physical level the last step on the logical level of the TIJAH system is a translation to Monet Interpreter Language (MIL). The MIL query plan is executed using MIL primitives that define the manipulation over Monet binary tables (BATs) [1].

The physical level is based on a pre-post numbering scheme [3] and the containment join operators ($\bowtie_{\sqsupset}$ and $\bowtie_{\sqsubset}$) introduced in [6]. In the specification of our retrieval model we first introduce three auxiliary functions at the physical level. These functions are used to compute the term frequency - $tf(r, R)$, the collection frequency - $cf(R)$ and the length prior - $lp(r)$. Variable λ represents the smoothing parameter for the inclusion of background statistics, μ is the mean value of the logarithmic distribution of the desired size for the element, and ρ is the variance (in our case set to 1) for the log-normal prior. These auxiliary functions can be implemented using two physical operators: a size operator $size(r)$ that returns the size of a selected region r, and a count operator $|R|$ that returns the number of regions in a region set R.

Function $tf(r, R)$ computes the term frequency of a term region set R, i.e., a set containing only regions representing a single term, in a region r, while function $cf(tm, R)$ computes the collection frequency of a term tm in the collection. They are computed as:

$$tf(r, R) = \frac{|R \bowtie_{\sqsubset} r|}{size(r)}, \quad cf(term, R) = \frac{|\sigma_{t=term, n=tm}(\mathcal{C})|}{size(Root)} \tag{1}$$

where tm is the name of a region $r \in R$, C denotes the set of all regions in XML collection, and $Root$ represents the region that is not contained by any other region in the collection (i.e., the region corresponding to the top node of the entire XML tree).

To define the length prior of the region r we used either the size of the element: $lp(r) = size(r)$, the standard element prior: $lp(r) = \log(size(r))$, or the log-normal distribution:

$$lp(r) = \frac{e^{-((\log(size(r))-\mu)^2/2\rho^2)}}{size(r)\rho\sqrt{2\pi}} \tag{2}$$

Although in our framework arbitrary implementations of abstract functions can be introduced on the physical level, for INEX 2004 we followed the language model [4] and the conclusion drawn from numerous experiments last year [6]. The abstract scoring functions defined in our region algebra, $f_{\sqsupset}(r, R)$ and $f_{\not\sqsupset}(r, R)$, implement the about function specified in NEXI, while $f_{\blacktriangleright}(r, R)$ and $f_{\blacktriangleleft}(r, R)$ specify the score propagation in nested regions:

$$f_{\sqsupset}(r, R) = \lambda tf(r, R) + (1-\lambda)cf(tm, R), \quad f_{\not\sqsupset}(r, R) = 1-(\lambda tf(r, R) + (1-\lambda)cf(tm, R)),$$

$$f_{\blacktriangleright}(r, R) = \frac{\sum_{r_i \in r \bowtie_{\sqsupset} R}(size(r_i) * p_i)}{\sum_{r_i \in r \bowtie_{\sqsupset} R} size(r_i)}, \quad f_{\blacktriangleleft}(r, R) = \sum_{r_i \in r \bowtie_{\sqsubset} R} p_i \tag{3}$$

In this paper we take the simple approach for the score combination operators where $\otimes$ is implemented as a product of two score values, and $\oplus$ as the sum of scores (with the default value $d = 1$), as it shows good behavior for retrieval.

3 Query rewriting

Upgrading the TIJAH system used previous year for INEX [6], we developed a fully automatic approach for translating NEXI queries, first into internal conceptual representation and later into logical algebra. The structure of the conceptual query translator is depicted in Figure 1. The conceptual level of the TIJAH system consists of three processing units: the query preprocessor, the

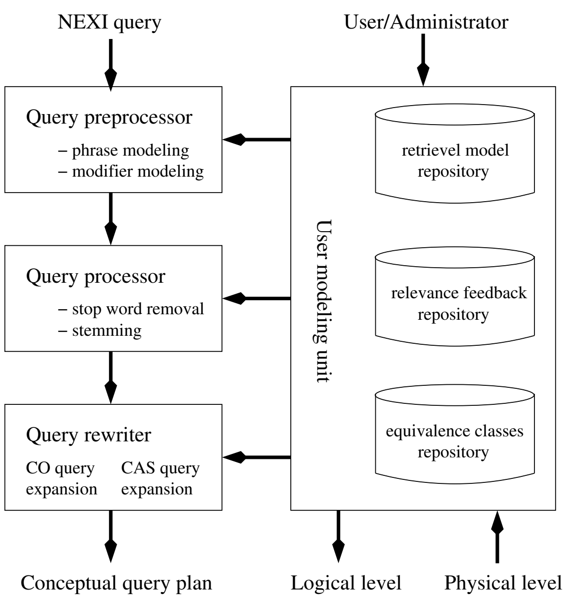

Fig. 1. The conceptual level of the TIJAH system.

query processor, and the query rewriter; and three user modeling units: the retrieval model repository, the relevance feedback repository, and the equivalence classes repository[4].

The retrieval model repository and relevance feedback repository form the input to the preprocessing unit. How the data from the repository is interpreted depends on a specification in the user modeling unit. The retrieval model repository in the TIJAH system currently only stores the parameters of a retrieval model, i.e., the smoothing parameter λ for the language model used for retrieval, and the desired size of the retrieved element for estimating the relevance on specificity.

Handling phrases and modifiers The query preprocessing unit rewrites NEXI queries based on a user specification on handling phrases and modifiers. Thus, users can specify whether phrases should be considered as phrases or as a set of terms in the query, and whether term and phrase modifiers ('+', and '−') should be considered during the query execution. The result of our example CAS query with phrases can be seen in Figure 2 as CAS query 1.

Stop word removal and stemming The standard IR query processing consisting of query stop word removal and stemming is done by the processing unit. We used the standard Porter stemmer and a publicly available stop word list consisting of 429 stop words. Stemming and stop word removal are applied based on a user specification (in the user modeling unit). In Figure 2 CAS query

[4] The equivalence classes repository represents a repository that should support retrieval from heterogeneous collections and is still not fully supported in the TIJAH system.

CAS queries:

```
1. ROOT//article[ABOUT(.//(abs|kwd),genetic algorithm
       "genetic algorithm")]//bdy//sec[ABOUT(.,simulated annealing)]
2. ROOT//article[ABOUT(.//(abs|kwd),genet algorithm "genet algorithm")]
       //bdy//sec[ABOUT(.,simul anneal)]
3. ROOT//article[ABOUT(.//(abs|kwd),genet algorithm "genet algorithm")
       AND ABOUT(.,genet algorithm "genet algorithm")]
       //bdy//sec[ABOUT(.,simul anneal)]
4. ROOT//article[ABOUT(.//(abs|kwd),genet algorithm "genet algorithm")
       AND ABOUT(.,genet algorithm "genet algorithm")
       AND ABOUT(.,simul anneal)]//bdy//sec[ABOUT(.,simul anneal)
       AND ABOUT(.,genet algorithm "genet algorithm")]
```

CO queries:

```
1. ROOT//article//*[ABOUT(.,+tree +edit +distance +"tree edit distance"
       +XML -image)]PRIOR
2. ROOT//article[ABOUT(.,+tree +edit +distance +"tree edit distance"
       +XML -image)]//*[ABOUT(.,+tree +edit +distance
       +"tree edit distance" +XML -image)]PRIOR
3. ROOT//journal[MATCH(tp,0.34)]//*[MATCH(sec,0.22) or MATCH(p,0.18)]
       [ABOUT(.,+tree +edit +distance +"tree edit distance"
       +XML -image)]PRIOR(856)
```

Fig. 2. The conceptual query representations generated from the NEXI queries.

2 depicts the outcome of the processing unit in case stemming would have been performed.

Query expansion The last step in conceptual query processing is query rewriting and expansion. The conceptual query rewriting unit distinguishes between NEXI content-only (CO) and content-and-structure (CAS) queries.

NEXI CO queries are transformed into CAS queries according to the user specification. For instance, the CO query can be translated in two ways:

– we are looking for any relevant XML element in any of the articles in the collection, including the articles themselves, as depicted in CO query 1 in Figure 2, [5] or
– we are looking for any relevant element in an article (including article elements themselves) that is about the topic specified, as depicted in CO query 2 in Figure 2.

The `PRIOR` in the conceptual query representation denotes that we use the relevance on specificity (i.e., result element size) in computing the final score of XML elements. The default is a length prior while for the log-normal prior the mean size should be specified (see CO example 3 in Figure 2).

Since NEXI CAS queries specify the element that should be retrieved as a result, query rewriting can only be about structural constraints in the *about* clause and about term distribution in different *about* clauses. Therefore, we applied two simple rules to enable elementary CAS query rewriting:

[5] In the TIJAH system, `//*` is treated as `descendant_or_self::node`.

– relaxing the constraint that terms or phrases must be contained by the XML elements specified in the structural part of the *about* clause, as depicted in CAS query 3[6] in Figure 2;
– further relaxing the structural constraints and allowing that terms or phrases in each subquery are also added to the other subqueries (similarly as we had in the TIJAH 2003 approach [6]), as shown in Figure 2, CAS query 4.

4 Phrase Modeling

For phrase modeling we follow the ideas introduced by Song and Croft [9], where authors individualized unigram and bigram language models and combined them in an independent way. We slightly modified their approach and used two interpretations:

– combination of n-gram LMs is modeled as an equally weighted sum: $P(t_1,..,t_{n-1},t_n|e) = P_1(t_1|e) \ + \ P_2(t_1,t_2|e) \ + \ ... \ + \ P_n(t_1,t_2,...,t_n|e)$, and
– combination of n-gram LMs is modeled as a product: $P(t_1,..,t_{n-1},t_n|e) = P_1(t_1|e) \times P_2(t_1,t_2|e) \times ... \times P_n(t_1,t_2,...,t_n|e)$.

In our approach the expression $P_i(t_1,t_2,...,t_i|e)$ gives the probability that the XML element e contains the phrase "t_1 t_2 ... t_i".

To be able to model phrases in region algebra we had to extend SRA to support phrase specification and more advanced score manipulation operators. For such a purpose we introduced a complex selection operator and additional containment operator defined in Table 2.

Table 2. Complex selection and containment operators.

Operator definitions
$\sigma_{t=type,n=name_1 \ adj \ n=name_2 \ adj \ ... \ adj \ n=name_n}(R) = \{r \mid (r_1 \in R \wedge r_2 \in R \wedge t_2 = t_1 = type \wedge n_1 = name_1 \wedge n_2 = name_2 \wedge e_1 = s_2 - 1 \wedge (s,e,n,t,p) := (s_1,e_2,n_2,adj,p_2)) \vee ... \vee (r_1 \in R \wedge r_2 \in R \wedge \ ... \ \wedge r_n \in R \wedge t_n = ... = t_2 = t_1 = type \wedge n_1 = name_1 \wedge n_2 = name_2 \wedge ... \ \wedge n_n = name_n \wedge e_1 = s_2 - 1 \wedge e_2 = s_3 - 1 \wedge \ ... \wedge e_{n-1} = s_n - 1 \wedge (s,e,n,t,p) := (s_1,e_n,n_n,adj,p_n))\}$
$R_1 \sqsupset_{\odot} R_2 = \{r \mid r_1 \in R_1 \wedge (s,e,n,t) := (s_1,e_1,n_1,t_1) \wedge p := p_1 \times f_{\sqsupset_{\odot}}(r_1,R_2)\}$

The first operator makes a union of all adjacent sub-regions in the region set that have the same type attribute. The name attributes of these new regions take the names of the regions from the second operand, while their type is now changed to `adj`. Following this approach and using *xp* as a shorthand for any arbitrary SRA expression formed during the NEXI to SRA query translation, phrase ˋtexttttree edit distance" can be transformed into the following logical expressions:

$$xp \sqsupset_{\odot} \sigma_{t=term,n='tree' \ adj \ n='edit' \ adj \ n='distance'}(R)$$

[6] Note that the stemming has been applied in the query processing.

Here, xp denotes an arbitrary SRA expression formed during the NEXI to SRA query translation.

The operator $\sqsupset_\odot$ defines how relevance scores from regions with distinct region name attributes (i.e., regions with adjacent terms of different length) are combined to form the resulting score value for regions in the left operand ($r_1 \in R_1$). The definition of function $f_{\sqsupset_\odot}$ is similar to the definition of function $f_\sqsupset$ in Equation 3 except that it treats adjacent regions with different name attributes in isolation and combines them based on the specifications of the operators $\otimes$ and $\oplus$, i.e., multiplying or summing them, respectively.

Furthermore, the scaling operator ($\circledast$) is introduced in the SRA to model terms with and without modifier '+'. The operator definition is as follows: $R_1 \circledast num = \{r | r_1 \in R_1 \wedge (s, e, n, t) := (s_1, e_1, n_1, t_1) \wedge p := (p_1 * num)\}$. This operator scales down the terms (regions) that are not marked with '+' and, therefore, considered not so important, and scales up terms (regions) marked with '+' that are considered important terms. In our approach important terms are scaled with a *num* value that is double the *num* value for not so important terms.

Based on the defined operators we can now translate the conceptual query representation into the logical query plan. For example, if we consider the first part of query 1 in Figure 2:

```
ROOT//article[ABOUT(.//(abs|kwd), genetic algorithm "genetic algorithm")]
```

we can express it in SRA as:

$$\texttt{ARTICLE} \blacktriangleright ((((\texttt{ABS} \sqcup_p \texttt{KWD}) \sqsupset_p \texttt{GENETIC}) \circledast 0.5) \sqcap_p$$
$$(((\texttt{ABS} \sqcup_p \texttt{KWD}) \sqsupset_p \texttt{ALGORITHM}) \circledast 0.5) \sqcap_p (((\texttt{ABS} \sqcup_p \texttt{KWD}) \sqsupset_\odot \texttt{GEN\_ALG}) \circledast 0.5))$$

Here we used ABS instead of $\sigma_{t=entity,n='abs'}$, KWD instead of $\sigma_{t=entity,n='kwd'}$, GENETIC instead of $\sigma_{t=term,n='genetic'}$, ALGORITHM instead of $\sigma_{t=term,n='algorithm'}$, and GEN_ALG instead of $\sigma_{t=term,n='genetic'\ adj\ n='algoithm'}$.

For the phrase modeling on the physical level we implemented two variants of the function $f_{\sqsupset_\odot}$, based on [9] and functions in Equation 1, as defined below:

$$f_{\sqsupset_\odot}(r, R) = \prod_{R_i} \eta tf(r, R_i) + (1 - \eta) cf(R_i) \tag{4}$$

$$f_{\sqsupset_\odot}(r, R) = \sum_{R_i} \eta tf(r, R_i) + (1 - \eta) cf(R_i) \tag{5}$$

Here, R_i is a set of regions (r_i) that have the same region name attribute (n_i). The sum and the product is defined over all sets R_i with different region names in R. Parameter η is used to specify the influence of foreground and background statistics for different adjacent regions in the region set.

5 Relevance feedback

Our approach for the relevance feedback track is based on the idea that knowledge of relevant components provides implicit structural *hints* that may help improve the performance of the content-oriented queries. We use a two-step procedure to implement this idea:

– First, we extract the *structural relevance* of the top-ranked elements according to the relevance assessments provided by the test collection;
– Second, the content-oriented query is rewritten into a structured one and the priors of the system are tuned based on the relevance feedback information. Then, the new structured query is evaluated in the TIJAH system.

In the following subsections, we define these two steps and explain their implementation in the TIJAH XML-IR system.

5.1 Extracting structural information

How to extract the *structural relevance* from the top relevant elements is a difficult problem. The semantics of the relevance assessments should be analyzed in depth to decide which type of *structural hints* should be extracted from the different relevant components and to define what is the best interpretation for the different relevance combinations.

In our first attempt to model the *structural relevance* of a query, we use the *journal* and the *XML tag* name information from the top-ranked elements as well as their relevance values. We also use the size of these elements to update the length prior for the next iteration in the relevance feedback cycle. We believe that with a good combination of these *hints* we can considerably improve the performance of the content oriented results. In Section 6 we give a preliminary analysis of the results of this approach.

The reminder of this section details how the structural information from the top-ranked elements and the results from relevance assessments on exhaustivity and specificity are used to rewrite the query for the next iteration in a relevance feedback process.

Journal name The content of the INEX collection consists of eighteen different journals. Each of these journals contains articles discussing a different computer science related field. We believe that when a component is assessed as relevant for a given topic, the journal it belongs to will contain elements with a similar content information. Therefore, we want to use this information to give a prior to the elements that are contained in that journal.

As an example, consider the relevance assessments for this year. If we consider only highly exhaustive and highly specific elements, marked with (3,3), we find that most of the topics have less than 3 relevant journals. That means, that likely, these are the journals that discuss the topic. Therefore, it is easy to imagine that other elements from these journals will also contain relevant information for that specific topic.

We decided to model the *journal prior* information according to the following formula:

$$P(J) = a + b \cdot \frac{\sum_{r \in top_{20} \sqsubseteq J} E_r}{3 \cdot |\{r \in top_{20} | E_r > 0\}|} + (1 - a - b) \cdot \frac{|J \sqsupseteq top_{20}|}{20}, \qquad (6)$$

where E_r is the exhaustivity value of the relevant top 20 components ($r \in top_{20}$) that belong ($\sqsubseteq$ and $\sqsupseteq$, respectively) to the journal J and a and b are weighting parameters used to tune the importance of this information.

Note that we only use the exhaustivity information to get a prior for the journal. We argue that if a component is somewhat exhaustive, it means that the journal it belongs to is likely to be about the topic need, i.e., to contain the desired information specified in the query (whatever the specificity for that component is). We also reward the journals that have a higher number of elements in the top 20 ranked elements (see the third part of the sum in the equation).

XML tag names The goal of using the element names for relevance feedback is to push up in the ranked result list the kind of elements we already know to be relevant for the topic ("make sense to be retrieved") and to push down the ones that are not.

For modeling the information on the XML tag names extracted from the top-ranked elements (e), we use a similar approach as for the journals (Equation 6):

$$P(e) = a + b \cdot \frac{\sum_{r \in top_{20} \sqsubseteq e} E_r + S_r}{6 \cdot |\{r \in top_{20} | E_r \cdot S_r > 0\}|} + (1 - a - b) \cdot \frac{|e \in top_{20}|}{20} \tag{7}$$

In this case, we also take into account the *specificity* scale (S_r) as it gives information on the size of the element: i.e, if the element was large enough or too small for the information need.

Size We use the size information to tune the length prior of our retrieval model for the next iteration. We believe that elements similar in length to those that are assessed as relevant have a higher likelihood to be the ones that the user is looking for. Nevertheless, it is not easy to combine the sizes of the top components to estimate a *desired* size to be retrieved.

We decided to use the following formula to define the *desired size* given the elements (r) in the top 20:

$$DesiredSize = \frac{\sum_{r \in top_{20}} size(r) \times SizeModifier_r}{\sum_{r \in top_{20}} sgn(SizeModifier_r)}, \tag{8}$$

where $\mathit{SizeModifier}_r$ is defined as:

$$\mathit{SizeModifier}_r = \begin{cases} 1 & \text{if } (E_r, S_r) \in \{(2,2),(3,3)\} \\ 0 & \text{if } (E_r, S_r) \in \{(1,1),(0,0)\} \\ \frac{3 - E_r + S_r}{3} & \text{otherwise} \end{cases} \tag{9}$$

We based this formula on the assumption that very specific components that are not very exhaustive (i.e., $S_r > E_r$) are likely to be too small to answer the information need and, on the other hand, highly exhaustive components that are not very specific (i.e., $E_r > S_r$) are likely to be too large as an answer. In case there are no relevant elements in the 20 top-ranked elements, or the relevant elements are marginally exhaustive and marginally specific, i.e., marked with (1,1), we use a default value for desired size. We also experimented with the following, simpler, estimation of the $\mathit{SizeModifier}_r$ than in Equation 9:

$$\mathit{SizeModifier}_r = \begin{cases} 0 & \text{if } (E_r, S_r) \in \{(1,1),(0,0)\} \\ \frac{S_r}{E_r} & \text{otherwise} \end{cases} \tag{10}$$

5.2 Rewriting and evaluating the query

After the information extraction on the structural information of the relevant elements and its fusion with the relevance assessments on exhaustivity and specificity, in the first step, the information is stored in the relevance feedback repository[7]. The information is used to rewrite the CO query and evaluate it in the TIJAH system. Assuming that in the relevance feedback repository for topic 166 the journal name specification is `tp` with a relevance prior 0.34, element names are `sec` and `p` with relevance prior values 0.22 and 0.18, respectively, and estimated element size is 856, the conceptual query after rewriting will look like the query given in Figure 2 as CO query plan 3. The `MATCH(e_name,imp)` is used to denote that the elements in the `e_name` have the higher probability to be relevant answers to a query than other elements, and the value `imp` gives their respective relevance prior.

The `MATCH` expression in the conceptual query representation is translated into a combination of a selection and scaling operators on the logical level. For example, query excerpt:

```
xp[MATCH(sec,0.22) or MATCH(p,0.18)]
```

is expressed on the logical level as:

$$xp \ \sqcap_p \left((\sigma_{n='sec',t=entity}(C) \circledast 1.22) \sqcup (\sigma_{n='p',t=entity}(C) \circledast 1.18)\right)$$

where C is the collection of all regions (all XML elements and terms in the collection). These operators are further translated into a physical query plan as defined in the previous sections.

6 Experiments

In this section we give an overview of the experiments we did with the TIJAH XML-IR system and present our results (official and additional runs) for the ad-hoc retrieval task (CO and CAS) and the relevance feedback task.

6.1 INEX ad-hoc track

This year we used a completely new implementation on the physical level of our system, we designed different experiments to evaluate which would be the best parameters for the retrieval model as well as to check if the scenarios used in previous years would produce the same performance on the new implementation.

CO queries For the CO task we designed two main experiments: The first one evaluates the effect of supporting phrases in the TIJAH XML-IR system as explained throughout the paper. Results of these runs are shown in Table 3. The different columns show the different approaches used to model the phrase

[7] Currently the computation and specification of these values are not completely integrated into the TIJAH system.

Table 3. CO experimentation runs: basic language model without length prior. Effects of supporting phrases. The 'n-gram' column indicates the kind of combination for the n-gram LMs and the $\odot$ column indicates the way the scores within a region are combined for a final score (Equations 4 and 5).

Run	λ	**n-gram**	$\odot$	**avg. MAP**	**overlap**
$R_{comp_{nophr}}$	0.35	-	-	0.0446	49.4%
$R_{comp_{phr1}}$	0.35	product	product	0.0437	51.1%
$R_{comp_{nophr}}$	0.5	-	-	0.0496	52.3%
$R_{comp_{phr2}}$	0.5	product	sum	**0.0502**	82.8%
$R_{comp_{phr3}}$	0.5	sum	sum	0.0470	82.1%

Table 4. Additional CO experimentation runs: Length priors

Run	**length prior**	**Avg MAP**	**overlap**
$R_{comp_{05}}$	none	0.0496	52.3%
$R_{comp_{cut5}}$	$res > 5$	0.0534	51.8%
$R_{comp_{cut10}}$	$res > 10$	0.0578	53.2%
$R_{comp_{cut25}}$	$res > 25$	0.0635	55.4%
$R_{comp_{cut50}}$	$res > 50$	0.0645	57%
$R_{comp_{logn}}$	log normal	0.0526	51.1%
$R_{comp_{logs}}$	log standard	**0.0985**	73.6%

search (see Section 4 for details). According to the results, supporting phrase search improved the retrieval performance in only one of the runs. Note that this improvement is partially positive as the overlap increased considerably too.

The second experiment was designed to evaluate the effect of including a length prior in the retrieval model. We defined several priors and applied them to the best of our runs. The results are shown in Table 4. In the first four runs, the length prior consists on removing from the result list the elements smaller than a certain threshold. The last two runs use a log normal and a log standard distribution to model the length prior. We can see that, whatever the prior is, the performance of the original run improves. As we saw already in previous years, a log standard distribution works best in our case, reaching a MAP of 1.4 in one of the metrics.

VCAS task For the VCAS task, we designed three different scenarios. The first one, R_{strict} treats the structural constraints of a query strictly and all the result components must exactly match these structural constraints. The second and the third one, R_{relax} and R_{all}, implement the relaxations of the structural constraints explained in Section 3, and shown in queries 3 and 4 in Figure 2. The results of these runs are shown in Table 5. Even if the improvement for the first relaxation (second row) is not significant, we can see in the precision and recall graphs that the relaxation of the structural constraints leads to better precision for this run. Contrary to last year, the second relaxation by using all the terms in all the about clauses did not performed as expected. Further analysis should

Table 5. Official VCAS experiment runs. Note that results in R_{all} have been modified due to an implementation error in the submitted ones

Run	**Avg MAP**	**overlap**
R_{strict}	0.0624	22.8%
R_{relax}	**0.0694**	24.3%
R_{all}	0.0588	23.9%

determine if this is just an effect of the different topics for this year or if, in general, the relaxation is not appropriate for our purposes.

6.2 INEX relevance-feedback track

The aim of the experiments submitted for the relevance feedback task is to identify which combination of the different structural and size information works better. The MAPs obtained with the different combinations experimented and their results are shown in Table 6.

Table 6. Official Relevance Feedback runs. Note that baseline runs are the result of removing the descendants of the top 20 elements from the original CO runs. Relevance feedback is applied on the residual collections of these runs, after freezing the top 20 elements. S1 and S2 refer to the size information extracted using Equations 9 and 10, respectively, J refers to journal information and XT to XML tag information.

baseline	S1	S1+J	S1+XT	S2	S2+J	J	S1+J+XT	XT	XT+J
0.0405	0.0406	**0.0416**	0.0406						
0.0431				0.0429	0.0448	**0.0450**			
0.0456							0.0482	**0.0486**	0.0468

The results show that none of the combinations improves significantly the performance of our system. Further experimentation is required to see whether different values for the parameters of the formulas presented will give a better performance or some other interpretation of the relevance assessments should be done.

7 Conclusions and Future Work

Our participation in INEX is characterized by applying a fully systematic approach able to support different retrieval tasks identified as ad-hoc tasks (CO and CAS) with simple user modeling and relevance feedback. We investigated the influence of phrases in the retrieval model with respect to the retrieval effectiveness. Furthermore, we experimented with straightforward approaches to (blind) *structural* relevance feedback. Future research includes more extensive experimenting in the area of phrase search and relevance feedback, applying new models for incorporating different aspects of relevance feedback information, and taking more advanced methods for phrase search, by adapting IR approaches such as classifier-thing bigrams [5], by using the WWW as N-gram training data [12], by using vocabulary clustering [8], etc., to XML phrase modeling. Finally, we aim to improve the efficiency of the system on, both memory and CPU wise, using rewriting and optimization rules on the logical level as well as by applying horizontal fragmentation and encoding of the data into more compact structures on the physical level.

References

1. P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications*. PhD thesis, CWI, 2002.
2. 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.
3. T. Grust and M. van Keulen. Tree Awareness for Relational DBMS Kernels: Staircase Join. In H. M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, editors, *Intelligent Search on XML*, volume 2818 of *Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, pages 179–192. Springer-Verlag, Berlin, New York, etc., August 2003.
4. D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Twente, The Netherlands, 2001.
5. M. Jiang, E. Jensen, S. Beitzel, and S. Argamon. Choosing the Right Bigrams for Information Retrieval. In *Proceeding of the Meeting of the International Federation of Classification Societies*, 2004.
6. J. List, V. Mihajlović, A. de Vries, G. Ramirez, and D. Hiemstra. The TIJAH XML-IR System at INEX 2003. In *Proceedings of the 2nd Initiative on the Evaluation of XML Retrieval (INEX 2003)*, ERCIM Workshop Proceedings, 2004.
7. 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.
8. R. Rosenfeld. Two Decades of Statistical Language Modeling: Where do we go from here? In *Proceedings of the IEEE*, 2000.
9. F. Song and W. B. Croft. A General Language Model for Information Retrieval. In *Proceedings of the eighth international Conference on Information and Knowledge Management*, pages 316–321, 1999.
10. 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)*, ERCIM Publications, 2004.
11. M. van Keulen. Relational Approach to Logical Query Optimization of XPath. In *Proceedings of the 1st Twente Data Management Workshop (TDM'04) on XML Databases and Information Retrieval*, pages 52–58, 2004.
12. X. Zhu and R. Rosenfeld. Improving Trigram Language Modeling With the World Wide Web. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 533–536, 2001.