TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback*

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

[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, 1090 GB Amsterdam, The Netherlands
{georgina, thijs, arjen}@cwi.nl

Abstract. Retrieving information from heterogeneous data sources in a flexible manner and within a single (database) framework is still a challenge. In this paper we present several extensions of our prototype database system TIJAH developed for structured retrieval. The extensions are aimed at modeling vague selection of XML elements and image retrieval. All three levels (conceptual, logical, and physical) of the TIJAH system are enhanced to support the extensions. Additionally, we analyze different ways of removing overlap and explain how structural information can be used for relevance feedback.

1 Introduction

In this paper we discuss our participation at INEX 2005 with TIJAH, a three-level database system for structured information retrieval. The TIJAH system [12, 13, 15] is developed as a transparent XML-IR database system consisting of conceptual, logical, and physical levels. TIJAH was originally developed to handle queries with the strict selection of XML elements, specified in the NEXI query language [18] and to reason about textual information. This year we extended it in three directions: toward handling vague specification of XML elements in the query (similar to [5]), toward supporting retrieval from heterogeneous domains (images and videos), following the guidelines from multimedia retrieval database systems [3], and toward supporting different approaches to remove overlap. Moreover, we continue with the relevance feedback experiments [16] using the TIJAH system.

The first point that we want to address in this paper is handling imprecise specification of elements in the XML search. Similarly to users giving a number of terms as hints for searching within a document, XML elements specified

* The research described in this paper is funded by NWO grant number 612.061.210.

within the query need not be considered as a strict requirement but as a hint for structural search. Therefore, when formulating a query the user can state that the search (support) element or answer (target) element should be treated as a hint or as a constraint in the retrieval process. To support vague search we introduced *vague element selection* as a concept in our TIJAH system.

On the other hand, to cope with the heterogeneous data sources (text and images) each level of the TIJAH system is extended with new features that can express *image search*. Image search is handled in the same framework as text search: At the conceptual level where NEXI query language is extended for query by example image search, and at logical level where new operators are introduced in the Score Region Algebra (SRA) [13]. However, due to different nature of the domain data, images are stored and handled in a different manner than textual XML data at the physical level.

We also present our approaches for removing overlap and for relevance feedback. To remove overlapping elements from the result set (for the user not to see the same information twice), we define a *utility* function that intends to capture the amount of *useful* information each element contains. Once we know the *utility* value of each node, we remove overlap by returning the most *useful* node in each path. Our relevance feedback approach uses the structural characteristics of the relevant elements to update the priors in a language modeling framework.

The paper is organized as follows. The following section explains the extensions introduced in the TIJAH system to model vague XML element specification. Section 3 details our approach for image retrieval. The overlap and relevance feedback approaches are discussed in Section 4 and 5 respectively. We wrap-up the paper with the results from the experiments performed for each track and its sub-tasks in Section 6 and with conclusions and future directions in Section 7.

2 Vague Node Selection

This section details the motivation and the implementation of vague selection of nodes in our three-level database framework. We explain the extensions on each level aimed for vague search on elements.

2.1 Vague element node selection in NEXI

Instead of extending our conceptual parser for rewriting content-and-structure (CAS) and content-only plus structure (COS) queries into SV, VS, and VV CAS and COS queries (SSCAS and SSCOS are equal to CAS and COS in our case), where prefix 'S' denotes strict and 'V' vague specification of target and support elements, we decided to extend the NEXI grammar with one extra symbol '$\sim$'. The 'tilde' symbol is used in front of the element name in the query specification, denoting that the element name does not have to be strictly matched in the query evaluation. We support this decision by arguing that the user should be responsible for stating his confidence in the knowledge of the hierarchical

organization of the data he is querying, or whether he is certain or not what the element name is in which he wants to search for information.

Since we decided to extend the NEXI syntax with the vague selection we had to manually rewrite the queries for each CAS and COS scenario except the SSCAS and SSCOS. For example, the (SS)CAS query 225:

```
//article[about(.//fm//atl, "digital libraries")]
                    //sec[about(.,"information retrieval")]
```

is rewritten into three variants:

- SVCAS: `//article[about(.//~fm//~atl, "digital libraries")]`
  `//sec[about(.,"information retrieval")]`
- VSCAS: `//article[about(.//fm//atl, "digital libraries")]`
  `//~sec[about(.,"information retrieval")]`
- VVCAS: `//article[about(.//~fm//~atl, "digital libraries")]`
  `//~sec[about(.,"information retrieval")]`

We decided not to consider the 'article' element as a vague element in case it is not the target element or it is not the element in which the *about* search should be performed, as in these cases the 'article' element just serves as a focusing element for deeper search in the XML tree.

Vague element selection can be treated similarly as a query expansion on terms in traditional IR. For example, if a user searches for the term 'conclusion', he might also be satisfied with terms 'decision', 'determination', 'termination', or 'ending' in the answer. In structured documents, if a user asks for 'car' elements, he would probably not mind getting 'auto' or 'vehicle' elements as an answer. The problem of element name matching is studied in the research area of schema matching and numerous techniques exist that try to resolve this problem (see [4] for survey). However, we decided to simplify the vague element name search task and use the results from INEX 2004 assessments to find the expanded element names. We define the list of expanded element names based on the list of element names assessed as relevant in INEX 2004 assessments process. The lists that we exploit in this paper, termed *element name expansion lists* are the following[3]:

- One manual set of lists with the default score 0.55, based on 2004 experiments (for the complete lists see [14]). For example, `sec` expansion list looks like: {`sec, abs, fm, vt, p, article, bdy, bm, app`}. The lists are formed out of highly exhaustive elements in the assessments list and by making the lists symmetric in terms of adding the most useful IEEE collection element names, such as `sec,p,abs,...`, to the expanded name list of other element names that are in the particular expanded element name list. For example, since for the `abs` element name, the `kwd` element name is in its expanded list, `abs` is added to the `kwd` expansion list.
- One set of lists automatically generated out of assessments with marginal, fair, or high exhaustivity and specificity. The default score is based on a number of relevant elements of that specific name, normalized by a total number of relevant elements, for all distinct target elements. For example, if 5 out of 50 elements assessed as relevant for `sec` answer element in the 2004 assessments set are `p` elements than the default score for `p` elements is 0.1.

[3] A more exhaustive set of expansion lists can be found in [14]

2.2 A complex selection operator for vague node selection

The logical level is based on Score Region Algebra – SRA [13]. The SRA data model consists of a set of regions, each defined by its start (s), end (e), type (t), name (n), and score (p). The operators in SRA are selection operators ($\sigma_{n=name,t=type}(R)$, $\sigma_{\diamond num}(R)$, $R_1 \sqsupset R_2$, and $R_1 \sqsubset R_2$), score computation operator ($R_1 \sqsupset_p R_2$), score combination operators ($R_1 \sqcap_p R_2$ and $R_1 \sqcup_p R_2$), and score propagation operators ($R_1 \blacktriangleright R_2$ and $R_1 \blacktriangleleft R_2$). Retrieval models are transparently implemented using abstract functions for score computation ($f_{\sqsupset}$), combination ($\otimes$ and $\oplus$), and propagation ($\blacktriangleright$ and $\blacktriangleleft$) [13].

The vague node selection at the conceptual level (NEXI) is translated into complex vague node selection operator at the logical level. The operator is defined in SRA as a union of all XML element regions that match the names of the 'expanded name regions' within the element name expansion list. By default all 'expanded regions' are down-weighted by a predefined factor. The definition of the operator is as follows:

$$\sigma^{expansion(class)}_{n=name,t=type}(R_1) := \{(r_1.s, r_1.e, r_1.n, r_1.t, r.p) \mid r_1 \in R_1 \ \wedge \ r_1.t = type \\ \wedge \ (r_1.n, r.p) \in expansion(class, name)\} \quad (1)$$

Here $expansion(class)$ is a set that contains all the expansions for all the region names in one expansion class, where expansion list for each region $name$ is:

$$expansion(class, name) := \{(ex\_n_1, ex\_w_1), (ex\_n_2, ex\_w_2), ..., (ex\_n_n, ex\_w_n)\}$$

Here ex_n_i is a expanded element name and ex_w_i is a real number in the range $[0, 1]$ denoting the down-weight factor. The operator $\sigma^{expansion(class)}_{n=name,t=node}(R_1)$ assigns name (ex_n) and score (ex_w) values to the region name (n) and score (p) based on the name and score values in the expansion list $expansion(class, name)$.

For the vague selection we use the fusion of equivalence classes [11] (eq_class) and our manual and automatic INEX 2004 expansion element name lists. This is done in such way that every expanded element name in these lists that has the equivalent name in the eq_class $name$ part is also expanded with the eq_class equivalent names for $name$. This expansions are termed $manual55$ for manual run and mm for the automatic one. Therefore, the eq_class selection on section elements can be expressed as $\sigma^{expansion(eq\_class)}_{n=\text{'sec'},t=node}(R)$, and vague node selection ($\sim$`sec`), using manual expansion list, can be transformed into the next SRA operation $\sigma^{expansion(manual55)}_{n=\text{'sec'},t=node}(R)$. In such a way we can transparently define the set of expanded nodes and their respective weights and use them for vague node selection in a vague element name selection retrieval scenarios.

2.3 The implementation of the vague selection operator

At the physical level, since we are working with the known INEX IEEE data collection, and as we used static INEX equivalence element name list and expansion element name lists based on INEX 2004 assessments, we decided to replicate the lists and store them as tables at the physical level, i.e., in MonetDB [1]. Thus, we have three tables with *(entity_name, expansion_name, expansion_weight)* for

*manual*55, *mm*, and *eq_class* lists. The complex selection operator is then implemented as an additional MIL function (MIL stands for MonetDB Interpreter Language and is used to implement operators in TIJAH [15]) that uses data from these tables.

For example, the vague *name* selection operator on region table R and the 'expansion regions' table S for the *manual*55 element names in the expansion list, in relational algebra can be defined as:

$$\pi_{r.s,r.e,r.n,r.t,s.weight}(\sigma_{s.n=name}(S) \bowtie_{s.n=r.n} (\sigma_{r.t=node}(R)))$$

Retrieval models We based the instantiation of retrieval models on the best models used for flat-file information retrieval, as well as XML retrieval: language models [7], the Okapi/Inquery model [2, 17], and Garden Point XML (GPX) [6]. For the relevance score computation we used language modeling in most of the cases (Equation 2) and Okapi and GPX in some of them (details of these models can be found in [13]). We used sum for upwards ($f_{\blacktriangleright}(r_1, R_2)$) and downwards ($f_{\blacktriangleleft}(r_1, R_2)$) score propagation (Equation 3). Abstract score combination operators $\otimes$ and $\oplus$ are implemented both as simple sum, or as product and sum, except in the case of the GPX model where the instantiation is given in Equation 4. In Equations 2 to 4: $r_1 \prec r_2 \equiv r_1.s > r_2.s \wedge r_1.e < r_2.e$, $size(r) = r.e - r.s - 1$, and *Root* is the collection root region.

$$f^{\mathrm{LM}}_{\sqsupset}(r_1, R_2) = p_1 \cdot (\lambda \frac{\sum_{r_2 \in R_2 | r_2 \prec r_1} p_2}{size(r_1)} + (1 - \lambda) \frac{|R_2|}{size(Root)}) \tag{2}$$

$$f_{\blacktriangleright}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_1 \prec r_2} p_2 \ , \qquad f_{\blacktriangleleft}(r_1, R_2) = p_1 \cdot \sum_{r_2 \in R_2 | r_2 \prec r_1} p_2 \tag{3}$$

$$p_1 \otimes p_2 = p_1 \oplus p_2 = \begin{cases} p_1 + p_2 & \text{if } p_1 = 0 \vee p_2 = 0 \\ A \cdot (p_1 + p_2) & \text{otherwise} \end{cases} \tag{4}$$

3 Image similarity search

To enable search on multimedia collection (provided by Lonely Planet) we also introduced extensions to the TIJAH system defined along three levels of our prototype DB. Also, we extended the NEXI syntax with an extra token '`src:`' that defines the location of the source image with which the destination image should be matched. Therefore, in multimedia query 11:

```
//destination[about(.//image, fruit vegetables src:/images/BN2787_4.jpg)]
   //point_of_interest[about(., food fruit vegetable market)]
```

the first *about* contains a request for image similarity search. The destination image that need to be matched is `images/BN2787_4.jpg`. In the preprocessing step, the '`src:`' part of the *about* is transformed into *about_image* and its relative path given in the NEXI '`src:`' specification is resolved into the path to the location where the data for image matching is stored. The image about command is then forwarded to the logical level.

3.1 Image search in SRA

To express image search in SRA we extended the SRA operator set with the additional operators σ^i and $\sqsupset_p^i$. The σ^i operator has similar definition as basic score region algebra operator $\sigma_{n=name,t=type}(R)$, except that the score p is now computed by a call to an external function f^i. The function f^i uses information extracted from the sample image and the image that should be selected and it computes the score of an image region based on similarity between the sample image and the selected image:

$$\sigma_{n=name}^{i\approx sample}(R_1) := \{(r_1.s, r_1.e, r_1.n, r_1.t, f^i(r_2.n, sample) \mid r_1 \in R_1 \ \wedge \ \exists r_2 \in C \\ \wedge \ r_2 \prec r_1 \ \wedge \ r_2.t = attr\_val \ \wedge \ r_1.t = attr \ \wedge \ r_1.n = name\} \quad (5)$$

Here *sample* is the location of the sample image data specified with the '`src:`' statement in the NEXI query, resolved in the preprocessing step, C is a set of all regions in the database, *attr* is the attribute node, and *attr_val* is value of the attribute *attr*.

The operator $\sqsupset_p^i$ is defined in the same way as $\sqsupset_p$ operator (for the exact definition see [13]), except that it allows computing score of a region containing images with the usage of different scoring formula than for terms (e.g., given in Equation 2). Therefore, the *about_image* in the multimedia query 1 is transformed into the next SRA expression:

$$\sigma_{n='image',t=node}(C) \sqsupset_p^i \sigma_{n=file\_name}^{i\approx\text{'BN2787\_4.jpg'}}(C)$$

3.2 Implementation of image search

At indexing time, we estimated a generative probabilistic model of each of the images in the collection (see below); the model parameters are stored in separate tables in the database. In addition, we constructed a table that links the image identifiers to the corresponding nodes in the collection tree. The image selection operator is implemented as a new MIL function that computes the similarity between each collection image model and the example image.

Retrieval model Similarity between example images and collection images is estimated using Gaussian mixture models (GMM). To this end, each of the images in the collections ($\omega(n_i)$) is modeled as mixtures of Gaussians with a fixed number of components K:

$$P(\boldsymbol{x}|\omega(n_i)) = \sum_{k=1}^{N_K} P(K_{i,k}) \ \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_{i,k}, \boldsymbol{\Sigma}_{i,k}), \quad (6)$$

where N_K is the number of components in the mixture model, $K_{i,k}$ is component k of class model $\omega(n_i)$ and $\mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The score of an image node given an example image from a query, is determined by the likelihood that the corresponding model generates the feature vectors ($\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$) representing the example

image. Like in the LM case for text, we interpolate with a background model based on collection statistics:

$$f^i(n_i, sample) = \prod_{\boldsymbol{x} \in \mathcal{X}_{sample}} [\lambda \cdot P(\boldsymbol{x}|\omega(n_i)) + (1-\lambda) \cdot P(\boldsymbol{x})] \quad (7)$$

The feature space of the vectors $\boldsymbol{x}$ is based on the DCT coefficients[4] obtained from 8x8 pixel blocks. For details of the feature vectors and the GMMs, see [19]. Equation 7 is used for similarity computation in the image selection operator $\sigma^{i \approx sample}_{n=name}(R_1)$. In image containment ($R1 \sqsupset^i_p R2$) the result score is computed as a product of scores of the region in the left and region in the right operand.

4 Overlap removal

In an XML retrieval setting, to identify the most appropriate elements to return to the user is not an easy problem. IR systems have difficult task to find out which are the most exhaustive and specific elements in the tree and return only these to the user, producing result lists without overlapping elements. So far, most of the approaches presented to remove overlap consist of post-filtering the ranked lists in one way or another. Basically, by selecting the highest scored element from each of the paths.

This would be a good strategy if the retrieval model would consider, when ranking, not only the estimated relevance of the XML element itself but also its *usefulness* compared to other elements in the same path. However, since most retrieval models rank elements independently, it is not always the case that the highest scored element is the most appropriate unit to return to the user. Therefore, the strategies to remove overlap that rely too much in the retrieval model scores are not always the most effective (see Section 6.1).

In the approach presented in this paper, we define an *utility* function that intends to capture the amount of *useful* information each element contains. This function is equivalent to giving an *utility* prior to the retrieval model scores. Our goal is to help the retrieval model to give a better estimation of the *usefulness* of each node and, in consequence, gain effectiveness when removing overlap.

To model the *usefulness* of a node, three important aspects need to be considered: (1) the relevance score estimated by the retrieval model, (2) the size of the element, and (3) the amount of irrelevant information it contains. For example, if a highly relevant element is very small, the amount of *useful* information it carries is also small. Whereas if a not so high scored element is longer, the amount of *useful* information that the user will read is larger. Thus, the decision of which elements are most *useful* should be related not only to the retrieval model scores but also to its length. That is why length normalization techniques are also used in XML Retrieval [9]. Similarly, whether to return a certain element or to return some of its children should be decided according to the amount of *irrelevant* information the user will have to read if the parent is returned.

[4] Discrete Cosine Transform, captures both color and texture information

To implement these ideas, we define an *utility* function that estimates the *usefulness* of a node as the product of the amount of relevant information that element contains, the element's score, and its length. Formally, for each of the XML nodes (E), the *utility* value is estimated as:

$$U(E) = (1 - \frac{\sum_{i \in nrch(E)} size(i)}{size(E)}) \cdot P(E) \cdot size(E) \quad (8)$$

Where $P(E)$ is the estimated relevance score given by the retrieval model and $nrch(E)$ is the set of non relevant children of E. Those children in which the *amount of relevant information* (estimated as the product of the element's length and score, $P(E) * size(E)$) is lower than a threshold (*quality* threshold). This utility function is equivalent to giving a length prior to the elements, but instead of using the whole element's length as prior, we try to estimate the size of the relevant information contained in it.

5 Relevance feedback

The main idea of any relevance feedback strategy is to use the knowledge of relevant items to retrieve more relevant items. So far, research has concentrated on using content-related information from the known relevant elements. However, for XML retrieval the structural characteristics of the relevant elements might also play an important role. Following the lines of what we started last year [15], we investigate the potential of the structural information for this type of task and analyze if retrieving structurally similar elements improves retrieval effectiveness.

5.1 Structural information in relevant elements

We study two different aspects of the structure of documents that can help the retrieval system to discriminate between relevant and non relevant elements. Namely, the containing journal of an element (the journal where that element belongs to) and the element type. Table 1 shows the number of different journals and element types judged relevant per topic. If we compare these numbers to the total number of different journals (24) and different element types (187) contained in the new collection, we can see that the knowledge of which journals and element types are relevant for each of the topics is a very important piece of information that can help retrieval systems to perform a better search.

One way to use the knowledge of which structural characteristics are relevant for a certain topic is to increase the a priori belief in relevance of the elements that have the same structural characteristics. In this way, we use the information of which relevant journals and element types are found in the top 20, to

Table 1. Number of different journals and element types judged relevant per topic. Statistics taken from relevance assessments 2005 version 2. Average over 28 CO topics. All degrees of relevance are taken into account.

Type info.	**Avg.**	**Median**	**Max**	**Min**
Relevant journals	7.9	8	16	2
Relevant element types	34.4	34.5	73	9

calculate priors and increase the a priori belief in relevance of the elements that are contained in that journal or that are from that specific element type.

5.2 Updating priors in a language modeling framework

For our baseline experiments, we used statistical language models (see Section 2.3). Using Bayes' rule and assuming independence between query terms, the probability of an element E given a query Q can be estimated as the product of the probability of generating the query terms q_i from the element's language model and the prior probability of the element:

$$P(E|Q) \propto \prod_{q_i \in Q} P(q_i|E)P(E) \tag{9}$$

Typically, little prior knowledge about the probability of an element is available and either uniform priors are used, or $P(E)$ is taken to be related to the element's length (i.e.,long elements are assumed to be more likely to contain relevant information) (cf. [9]). However, once we have some information about relevant elements, for example from a user's relevance judgments, we can use this information to update the priors. From the judgments, we can discover the characteristics of relevant elements and update the priors in such a way that elements with similar characteristics are favored[5].

Therefore, once we get information about the structural characteristics of the relevant elements for a given topic, we define the priors for the journals and element types and use them to retrieve structurally similar elements. However, since in the top 20 we may not have seen all relevant journals or element types, there is the risk of assigning a prior equal to zero to element types or journals that do actually contain relevant information. To avoid this effect of relying too much on what is seen in the top 20, we interpolate $P(x(E)|rel)$ with the general probability of seeing elements from $x(E)$. Thus the prior becomes:

$$P_x(E) = \frac{\alpha P(x(E)|rel) + (1-\alpha)P(x(E))}{P(x(E))}, \tag{10}$$

where $x(E)$ identifies the journal (element type) to which E belongs, $P(x(E)|rel)$ is estimated as the fraction of relevant items belonging to the journal (element type) and $P(x(E))$ is the fraction of elements in the collection that belongs to that journal (element type).

6 Experiments

Among numerous tracks and scenarios specified for INEX 2005, we participated in the following: all CO and CAS ad-hoc track sub-tasks, multimedia track, interactive track, and relevance feedback track. Below, after introducing the metrics reported in the paper, we will explain in detail our approaches for each of these (sub)tasks.

[5] Strictly speaking $P(E)$ can no longer be called a prior, since it depends on the topic at hand.

Metrics The official INEX metrics for 2005 ad-hoc and relevance feedback track are based on extended Cumulative Gain (xCG) metrics [10]. The official metrics are: normalized xCG (nxCG), effort-precision/gain-recall (ep/gr), and extended Q and R[6]. The evaluation can be done either with the generalized or with the strict quantization. In this paper we report the evaluation results obtained with nxCG (also denoted as CG in Table 4) at various recall points: 10, 25, and 50 and mean average ep/gr. For multimedia track we report mean average precision (MAP) values.

Note that for any document cut-off value, say 10, it can be shown that, if strict quantization is used (or any other binary quantization), and overlap is not taken into account, and the total number of relevant elements is bigger than 10, then nXCG at 10 and precision at 10 give exactly the same results. However, if the number of relevant elements is smaller than 10 for some topics, then this might have a big impact on the measured performance.

For instance, IBM Haifa's run "SSCAS_no_phrase_no_plus" and Max Planck Institute's (MPI) run "MPII_TopX_SSCAS" have the same average precision at 10 over 4 topics with relevant elements: 0.225 for both runs (over topic 256, 260, 270 and 275). That is, on average 22.5% of the elements inspected in the top 10 is highly exhaustive and specific. However, for one of these 4 SSCAS topics (topic 270), only 1 relevant document is known. Because of this, the nXCG at 10 over the 4 topics is twice as high for MPI (0.450), which found the document in its top 10, as it is for IBM (0.225), which did not find it in its top 10. Apparently, a 100% gain in nXCG does not have to say much about the actual percentage of relevant items seen by the user. Precision at x is less sensitive to the total number of known relevant elements than XCG at x, and therefore defining the ideal recall base as needed for XCG is not really an issue for precision [8].

6.1 Ad-hoc track: CO queries

Thorough The aim of the *Thorough* retrieval strategy is to find all highly exhaustive and specific elements. Thus, to find all relevant information regardless of overlapping results. This year we submitted only two runs with the aim of using them as baseline runs for the other tasks and sub-tasks. Description and results for these two runs are given in Table 2. Although under the strict quantization there are not big differences between the two runs, under the generalized, one of the runs ($\lambda = 0.4$) outperforms the other considerably. We used this run as baseline for the rest of the CO experiments.

Table 2. CO.Thorough experiments with strict (^{S}) and generalized (^{G}) quantization.

Run id	**Description**	**nXCG[10]**	**nXCG[25]**	**nXCG[50]**	**ep/gr**
LMs_04_lp^S	LMs, $\lambda = 0.4$, lp	0.0923	0.0885	0.1020	0.0511
CO_LMs_trm_085^S	**LMs, $\lambda = 0.85$, lp**	0.0923	0.0855	0.0859	0.0490
LMs_04_lp^G	LMs, $\lambda = 0.4$, lp	0.2480	0.2433	0.2213	0.0795
CO_LMs_trm_085^G	**LMs, $\lambda = 0.85$, lp**	0.2161	0.1856	0.1839	0.0596

[6] http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv6.pdf

Table 3. CO.Focussed experiments with strict (^{S}) and generalized (^{G}) quantization.

Approach	**nXCG[10]**	**nXCG[25]**	**nXCG[50]**	**ep/gr**
Naive: select articles^S	0.0115	0.0154	0.0192	0.0116
Naive: select sections^S	0.0308	0.0417	0.0404	0.0136
Naive: select paragraphs^S	**0.1209**	**0.1538**	**0.1630**	**0.0654**
Common^S	0.0978	0.0927	0.1082	0.0509
Utility^S	0.1016	0.1373	0.1498	0.0561
Naive: select articles^G	0.1582	0.1226	0.1024	0.0443
Naive: select sections^G	0.1857	0.1753	0.1519	0.0588
Naive: select paragraphs^G	**0.2310**	**0.2197**	**0.2105**	**0.0764**
Common^G	0.2193	0.1909	0.1892	0.0728
Utility^G	0.2127	0.1919	0.1977	0.0742

Focused The aim of the *Focused* retrieval strategy is to find the most exhaustive and specific element in a path. Once the element is identified and returned, none of the remaining elements in the path should be returned. In other words, the result list should not contain overlapping elements. In our experiments for this task, we investigate the differences in terms of effectiveness between different approaches to remove overlap and evaluate the approach presented in Section 4.

To compare approaches, we implemented two already known ways of removing overlap: namely, the *naive* and the *common* approach. The *naive* approach filters out from the result list everything except one specific type of element (assuming that there is no overlap between elements of the same type). The *common* approach is implemented as follows: first, we select the highest scored element from the result list and remove its ancestors and descendants, then we take the second highest scored element and remove its ancestors and descendants, and we continue recursively until all elements from the result list have been either selected or removed. To evaluate our approach we use different *quality* thresholds. We observe that although there are no significant differences between the performance of the runs under the generalized quantization, under the strict one, the best improvements are achieved when the threshold is not very high, i.e., when elements are less punished for having irrelevant information. We only report the results of the best run overall with the threshold defined as the score of the element at position 375 (1500/4) in the original result list. Note that the score of this element is usually very small and many of the elements exceed this threshold. The results of all these runs are shown in Table 3.

For all measures and quantizations, the approach that performs the best is the one that retrieves only paragraphs. In general, for the *naive* approach, and as expected for a *focused* retrieval task, the longer the element, the worse the performance. However, it is somehow surprising that the retrieval of sections, which is also a rather focused unit, is not performing well. The *common* approach performs well but under the strict quantization is still far from the best run. This might be because our baseline contains a length prior that rewards longer elements. Therefore, when removing overlap, the longer elements, which probably are not the most exhaustive and specific on the path, are selected. Comparing the scores of the approach presented in this paper to the ones of the *common* run,

Table 4. INEX 2005 CAS and COS.Thorough experiments with different vague scenarios and rewriting techniques evaluated using nxCG at different recall points and ep/gr with generalized quantization.

	CAS				**COS.Thorough**			
Exp. class	**CG[10]**	**CG[25]**	**CG[50]**	**MAP**	**CG[10]**	**CG[25]**	**CG[50]**	**MAP**
eq_class	0.2799	0.2851	0.2644	0.05033	0.2677	0.2258	0.1787	0.03205
rw I	0.2687	0.2834	0.2645	0.04670	0.2715	0.2430	0.1894	0.03323
rw II	**0.3030**	**0.2977**	**0.2679**	**0.05476**	**0.2872**	**0.2467**	**0.1898**	**0.03409**
mm(sv)	0.2865	**0.2882**	**0.2626**	0.05219	0.2772	0.2333	0.1951	0.03657
man55(sv)	**0.3066**	0.2853	0.2419	0.05291	0.2727	0.2349	0.1972	0.03650
mm(vs)	0.2672	0.2658	0.2524	0.06749	0.2827	0.2499	0.2042	0.04283
man55(vs)	0.2316	0.2417	0.2391	0.06720	0.2751	0.2410	0.2060	0.04587
mm(vv)	0.2811	0.2728	0.2529	0.07062	**0.2912**	0.2580	0.2258	0.06060
man55(vv)	0.2545	0.2553	0.2428	**0.07296**	0.2851	**0.2585**	**0.2315**	**0.06872**
mm(vv)$^{\text{rwI}}$	0.2734	0.2641	**0.2603**	0.05899	0.2929	0.2686	0.2262	0.06168
man55(vv)$^{\text{rwI}}$	0.2427	0.2691	0.2469	0.06872	**0.3040**	0.2676	**0.2395**	**0.07168**
mm(vv)$^{\text{rwII}}$	**0.3092**	0.2815	0.2366	0.05760	0.2873	0.2492	0.2168	0.05878
man55(vv)$^{\text{rwII}}$	0.3005	**0.2943**	0.2556	**0.06896**	0.2956	**0.2688**	0.2262	0.06891

we see that under the strict quantization the re-ranking of the scores using the *utility* function does improve performance considerably (in terms of precision at high recall levels). That means that the utility function does help the retrieval model to make a better estimation of the most *useful* elements in the paths. Unfortunately, in all runs our approach is outperformed by the naive approach of selecting paragraphs. A possible cause of this could be that our approach rewards longer elements too much, but further analysis need to be done to test this an other hypothesis.

6.2 Ad-hoc track: CAS & COS.Thorough experiments

In our experiments with queries that use structure we aimed at comparing vague node selection approaches for all four query types (SS, SV, VS, and VV CAS and COS) with two query rewriting techniques that we used previous years for INEX [12, 15]. These rewriting techniques treat structural constrains as strict but mix the terms in different about clauses. In the first rewriting approach (rw I), all terms that are in different *about* clauses in the same predicate expression, and are not at the top level (i.e., not in `about(., term)` expression), are added to an extra top-level *about* clause in the same predicate expression. The second approach (rw II), is an extension of the first one, where not only the terms from non top-level *abouts* are added to the new *about*, but also all the terms from the other predicate, if there exists any, are added to the top-level *about* in each predicate.

We report the results using only VVCAS and COS.Thorough assessments as we wanted to test the approaches on the same assessments set. We present only the results using generalized quantization as the results using strict quantization lead to the same conclusions (see extensive set of experiments in [14]). The runs given in bold are the best ones for each series of experiments. In the first

set of experiments we test how much we can improve the effectiveness when using rewriting techniques. The first three rows in Table 4 show that rewriting techniques help; e.g., the rw II shows overall better scores, especially for early precision as can be seen in CAS runs, and it also has higher MAP values.

In the second set of experiments we replace the strict queries with the vague ones. The improvements are significant and they can go up to more than 100% (e.g., for the MAP in COS "*man55(vv)*" run). Clearly, vague element selection has higher MAP values than rewriting techniques, but in all CAS experiments it has lower precision at low recall points. This can indicate that rewriting techniques might be used as a precision tool, while vague element selection can be considered as a recall tool. Looking at different vague scenarios, namely SV, VS, and VV, and except for some early precision scores, VV runs seem to have the best performance. Therefore, "*mm(vv)*" and "*man55(vv)*" runs are used in combination with rewriting techniques for further experiments.

The third set of experiments confirms our assumption about the rewriting techniques as a precision and vague element search as a recall enhancement tool. As can be seen in Table 4 in most of the cases the combination of rw I and rw II rewriting techniques and manual and automatic vague element search improves early precision. However, not in all cases we managed to keep the MAP values, especially for the rw II combinations as can be seen in CAS runs.

6.3 Multimedia track: image queries

An important goal of our multimedia extension was to showcase and test the flexibility and extendibility of the SRA approach. In addition, we tested if using visual similarity can contribute to better results. To this end, we compared the multimedia queries discussed in Section 3 to similar queries with all image similarity clauses (`src:`) removed. The results of these two approaches using three different models for text search is given in Table 5. There exist differences between the models, but we did not find any improvement using visual similarity, in fact the best run uses only textual language models and is significantly better than its multimedia counterpart. We believe this is partially due to the nature of the collection and topics, but more research is needed to investigate if and how visual information can help to improve retrieval results in this collection.

Table 5. Results for MM track.

LM	**MAP**	**Okapi**	**MAP**	**GPX**	**MAP**
text only	0.2751	text only	0.2110	text only	0.2567
multimedia	0.2600	multimedia	0.2133	multimedia	0.2627

6.4 Relevance feedback track

To analyze the effects of using structural information in the relevance feedback process as described in Section 5, we designed two main experiments. The first one varies the values for α in Equation 10 to analyze the effects of assigning

different importance to the structural information found in the top 20. The values used are: 0.75, 0.5 and 0.25. This experiment is done on top of the baseline used in the rest of experiments, namely: CO_LMs_04_lp. The second experiment aims to identify which of the two types of structural information provides better improvement to the overall effectiveness of the IR system. Therefore, we fix the value of α in Equation 10 to 0.5 and analyze the gain obtained when using journal priors, element priors, and both priors at the same time. This experiment is done on top of one of our runs for the COS.Thorough task that uses the VVCAS approach explained in Section 2.

Since the official results show that very little gain is obtain for any of the runs, we do not report the numbers here. However, we observe that the journal prior seem to slightly improve recall and that there is no significant differences in performance when using different α's. The element prior seems to deteriorate the retrieval scores. We believe that this prior would perform much better when combined with a content-oriented query expansion, but further analysis and experiments need to be done in order to test this hypothesis.

7 Conclusions and Future Work

Throughout the paper we show that the TIJAH database system is flexible enough to incorporate advanced search techniques, such as vague element selection and relevance feedback, and search on heterogeneous data sources, such as a combination of images and text. For vague search, query rewriting techniques seem to be more suitable for obtaining higher precision at low recall points, while vague element selection is more suitable for higher average precision. Their combination however can boost the early precision, but it can also have negative influence on mean average precision. The simple image search model shows no improvements when combined with text search model. The approach presented to re-rank retrieval scores using an *utility* function seems to improve effectiveness when removing overlap. Unfortunately, this method does not outperform the simple approach of selecting the paragraph elements. Using only the structural characteristics of the elements in a relevance feedback process does not help retrieval performance in our case.

We plan to continue the experimental evaluation of different scenarios for search in structured documents: (1) the focused search using different *utility* functions to improve the effectiveness of overlap removal, (2) the vague element search with different assignment of non-uniform down-weighting factors and its combination with rewriting techniques, (3) the usage of structural relevance feedback in combination with content-oriented query expansion, and (4) the image search for improving retrieval results.

8 Acknowledgments

We would like to thank Roberto Cornacchia at CWI, Amsterdam, for providing the visual similarity code and for pre-processing the Lonely Planet images.

References

1. P. Boncz. *Monet: a Next Generation Database Kernel for Query Intensive Applications*. PhD thesis, CWI, 2002.
2. J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY Retrieval System. In *Proceedings of the 3rd DEXA Conference*, 1992.
3. A.P. de Vries. *Content and Multimedia Database Management Systems*. PhD thesis, University of Twente, Twente, The Netherlands, 1999.
4. A. Doan and A.Y. Halevy. Semantic Integration Research in the Database Community. *AI Magazine*, 26:83–94, 2005.
5. N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM TOIS*, 22(2):313–356, 2004.
6. S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In N. Fuhr, M. Lalmas, and S. Malik, editors, *Proceedings of the 3rd INEX Workshop*, volume 3493 of *Lecture Notes in Computer Science*, pages 276–291, 2005.
7. D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, Twente, The Netherlands, 2001.
8. D. Hiemstra and V. Mihajlović. The Simplest Evaluation Measures for XML Information Retrieval that Could Possibly Work. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, 2005.
9. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length Normalization in XML Retrieval. In *Proceedings of the 27th ACM SIGIR Conference*, pages 80–87, 2004.
10. G. Kazai, M. Lalmas, and A.P. de Vries. The Overlap Problem in Content-oriented XML Retrieval Evaluation. In *Proceedings of the 27th ACM SIGIR Conference*, 2004.
11. G. Kazai, M. Lalmas, and S. Malik. INEX'03 Guidelines for Topic Developments. In *Proceedings of the 2nd INEX Workshop*, ERCIM Workshop Proceedings, 2004.
12. 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 INEX Workshop*, ERCIM Workshop Proceedings, 2004.
13. 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*, 2005.
14. V. Mihajlović, D. Hiemstra, and H. E. Blok. Vague Element Selection and Query Rewriting for XML Retrieval. In *Proceedings of the 6th Dutch-Belgian Information Retrieval Workshop*, 2006.
15. 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*, volume 3493 of *Lecture Notes in Computer Science*, pages 276–291, 2005.
16. G. Ramírez, T. Westerveld, and A.P. de Vries. Structural Features in Content Oriented XML Retrieval. In *Proceedings of the ACM CIKM Conference*, 2005.
17. 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.
18. 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 2nd INEX Workshop*, ERCIM Publications, 2004.
19. T. Westerveld. *Using generative probabilistic models for multimedia retrieval*. Ph.d. thesis, University of Twente, Enschede, The Netherlands, November 2004.