Evaluating Structured Information Retrieval and Multimedia Retrieval Using PF/Tijah

Thijs Westerveld[1], Henning Rode[2], Roel van Os[2], Djoerd Hiemstra[2], Georgina Ramírez[1], Vojkan Mihajlović[2], and Arjen P. de Vries[1]

[1] CWI, Amsterdam, The Netherlands
[2] University of Twente, Enschede, The Netherlands

Abstract. We used a flexible XML retrieval system for evaluating structured document retrieval and multimedia retrieval tasks in the context of the INEX 2006 benchmarks. We investigated the differences between article and element retrieval for Wikipedia data as well as the influence of an elements context on its ranking. We found that article retrieval performed well on many tasks and that pinpointing the relevant passages inside an article may hurt more than it helps. We found that for finding images in isolation the associated text is a very good descriptor in the Wikipedia collection, but we were not very succesful at identifying relevant multimedia fragments consisting of a combination of text and images.

1 Introduction

CWI and the University of Twente collaborated again for INEX. This year, we participated in the Ad Hoc and Multimedia tasks. For both tasks, we relied on the PF/Tijah system [3], a system for flexible information retrieval from structured document collections. PF/Tijah integrates NEXI based IR functionality and full XQuery support.

In the Ad Hoc track we focused on three aspects. First, we studied whether element retrieval could do better than article retrieval. Second, we experimented with context weighting. Third, we looked at approaches to identify good entry-points in relevant articles for the AllInContext and BestInContext tasks.

For the multimedia track, we did not do any image processing, all our approaches are purely text and context based. For the Multimedia fragments task (MMfragments), we extended the Wikipedia collection with the metadata from the image collection. Also, we made sure any submitted result contained at least one image. We experimented with query variants based on just the title terms of the distributed topics, as well as with extending this with terms originating from castitle's or image examples.

The remainder of this paper is organised as follows. First, we introduce the PF/Tijah system in Section 2. Then, Sections 3 and 4 discuss our approaches and results for the Ad Hoc and Multimedia tracks. The paper ends with conclusions in Section 5.

N. Fuhr, M. Lalmas, and A. Trotman (Eds.): INEX 2006, LNCS 4518, pp. 104–114, 2007.

2 The PF/Tijah System

PF/Tijah is a research project run by the University of Twente with the goal to create a flexible environment for setting up search systems. By integrating the PathFinder (PF) XQuery system [2] with the Tijah XML information retrieval system [4] it combines database and information retrieval technology. The PF/Tijah system is part of the open source release of MonetDB/XQuery developed in cooperation with CWI Amsterdam and the University of München. The system is available from SourceForge.

PF/Tijah includes out-of-the-box solutions for common tasks like index creation, stemming, stopword removal, and result ranking for structured queries (supporting several retrieval models), but it remains the same time open to any adaptation or extension.

The PF/Tijah system has a number of unique features that distinguish it from most other open source information retrieval systems:

- It supports retrieving arbitrary parts of the textual data, unlike traditional information retrieval systems for which the notion of a document or fields need to be defined up front at indexing time. A query can simply ask for any XML tag-name as the unit of retrieval without the need to re-index the collection.
- The system allows complex scoring and ranking of the retrieved results by directly supporting the NEXI query language.

```
return
pf:tijah-query($root, "//html[about(.,IR DB)]//p[about(.,XML)]")
```

- PF/Tijah embeds NEXI queries as functions in the XQuery language. This way the system supports ad hoc result presentation by means of its query language. For instance, when searching for a special issue of a journal, it is easy to print any information from that retrieval result on the screen in a declarative way (i.e., not by means of a general purpose programming language), such as the special issue title, its date, the editors and the preface. This is simply done by means of XQuery element construction. As another example, we can formulate a query that performs a whole INEX run and gathers the results in the required output format:

```
for $topic in doc("/INEX/topics2006.xml")//inex_topic
let $result := tijah-query-id($c, $topic/castitle/text())
return
<topic topic-id=.{$topic/@topic_id}">
{ for $r in tijah-nodes($result) return
<result>
<file> { $r/name/@id } </file>
<path> { local:getINEXPath($r) } </path>
<rsv> { tijah-score($result, $r) } </rsv>
</result> }
</topic>
```

– PF/Tijah supports text search combined with traditional database querying, including for instance joins on values. For instance, one could formulate the difficult INEX topic 14 from 2002 in the following way:
Find figures that describe the Corba architecture and the paragraphs that refer to those figures. Retrieved components should contain both the figure and the paragraph referring to it.

```
let $doc := doc("inex.xml")
for $p in tijah-query($doc, "//p[about(.,corba architecture)]")
for $fig in $p/ancestor::article//fig
where $fig/@id = $p//ref/@rid
return <result> { $fig, $p } </result>
```

Recent developments in the PF/Tijah search module mainly concerned stability, scalability and performance. We can index the current Wikipedia collection in 25 to 30 minutes on a 64 bits machine with a 2Ghz Opteron processor and 8 Gb of memory running Fedora Core 6. Querying times are shown in the following table:

Simple article query `//article[about(.,X)]` (top 10, ranking only)	2 sec
Full INEX `//article[about(.,X)]` query (top 1500, INEX results format)	28 sec
Full INEX `//*[about(.,X)]` query (top 1500, INEX results format)	141 sec
Complete INEX run	

3 Ad Hoc Track

The characteristics of the Wikipedia collection differ considerably from the IEEE collection used before. This inspired us to test some ideas that seem in particular suitable for this new collection, but also to revisit some of the approaches that were successful on IEEE and to test how well these techniques work on a very different set of documents. We studied element vs. article retrieval, context weighting and the entry-point tasks, each of these is discussed in a separate subsection below, but first we discuss our general approach.

3.1 Approach

For all our submissions, we employed the PF/Tijah system. We indexed the entire Wikipedia collection, without removing any structural information. The index was stemmed and stopwords were removed. All our submissions are based on the XML variant of the unigram language modelling approach to information retrieval [4]. Elements are scored based on a mixture of foreground or document statistics and background or collection statistic. When we need to combine information from different levels in the hierarchy, for example for queries like `//article[about(.,X)]//*[about(.,Y)]` we use a product of the element scores. All elements always have a score greater than zero because of the background statistics. Therefore, the product based combination functions as a weak AND.

We believe it is useless to show the same information twice to a user, and thus removed overlap from all our runs. As a result, all our runs can be interpreted as submissions for the Focused task and that's how we evaluate them below. Our focused runs are constructed from (thorough) pure language modelling runs, by iteratively going down the ranked lists and removing all lower ranked ancestor and descendant of a retrieved element.

3.2 Element vs. Article Retrieval

The IEEE collection contains mainly lengthy documents where it makes sense to retrieve parts rather than the whole document. For the Wikipedia collection this is the case to a much lesser extend. Wikipedia documents tend to be short and focused. Moreover, a substantial amount of real Wikipedia queries has a corresponding Wikipedia document whose title matches the query exactly. This indicates that for many queries, an entire document may be a good answer. We investigated this by comparing document retrieval and element retrieval approaches. To this end, we ran two types of title only queries against our PF/Tijah system:

- `//article[about(.,$title)]`
- `//*[about(.,$title)]`

Where $title is replaced by the terms from the `<title>` field. The results for these runs on the Focused task are shown in Figure 1.

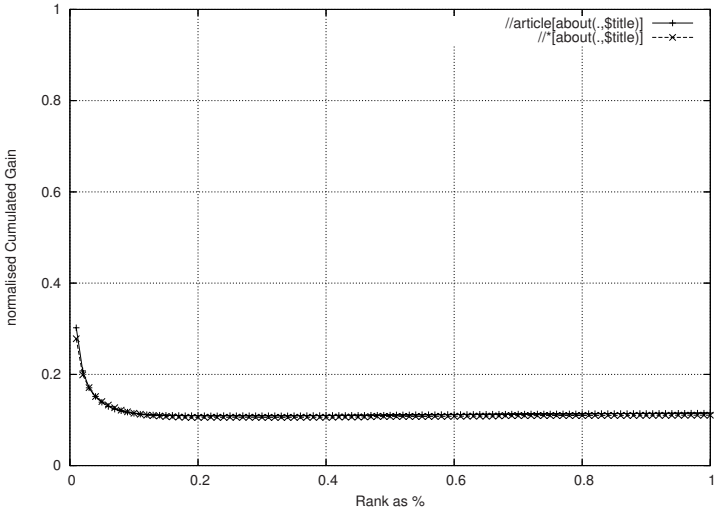

Fig. 1. Element vs. Article retrieval, normalised Cumulated Gain (overlap: on, quantisation: strict)

The runs are indistinguishable, indicating it makes no difference whether we retrieve full articles, or unrestricted elements. If we look closer at the retrieved elements, this makes sense. Figure 2a shows the element types we retrieved most in the element run. Almost half of the elements we retrieve are either *body* or *article*, and thus very close to full documents. Another 11% of the retrieved elements are of type *collectionlink*, and point indirectly to another article. We did not exploit this indirection, but these numbers indicate that effectively our element run was mainly pointing at full documents rather than smaller parts. This does not mean element retrieval is useless in this collection, though. Many of the relevant elements are of a much finer granularity, 45% of the relevant items are either paragraphs (*p*) or sections (*section*) (see Figure 2b).

32.5% body	32.7% p
17.3% p	12.4% section
16.3% article	9.0% item
11.2% section	8.4% emph3
11.0% collectionlink	6.3% emph2
Most retrieved element types with element run	Most relevant element types
a	b

Fig. 2. Element types. Most retrieved in element run (`//*[about(.,$title)]`) (a) and most relevant (b) element types

3.3 Context Weighting

Previous INEX results have shown that it is important to take an element's context into account. In particular article weighting has been successful [1,4]. Article weighting takes the article context of an element into account; good elements in good articles will be ranked higher than good elements in bad articles. We investigated whether article weighting is useful for element retrieval in the new Wikipedia collection.

In addition, articles in wikipedia have a clear and concise title, which may help in identifying relevant articles. We experimented with using this name together with the article content for article retrieval (and as a side-effect for setting the context for element retrieval).

We compared article retrieval with and without names[1]:

- `//article[about(.,$title)]`
- `//article[about(.,$title) OR about(.//name,$title)]`

And we experimented with article weighting:

- `//*[about(.,$title)]`
- `//article[about(.,$title) OR about(.//name,$title)]//*[about(.,$title)]`

[1] A limited preliminary study indicated that a disjunctive combination of article content and name performs better than a conjunctive combination.

Figure 3 and 4 show the results for name and article context respectively. Context and article weighting appear to have no influence on retrieval effectiveness. Thus context appears to be less influential than in the IEEE collection.

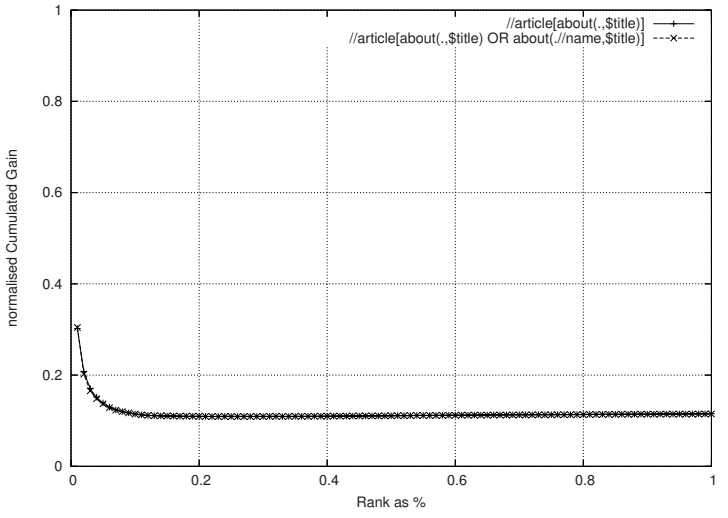

Fig. 3. Normalised Cumulative Gain for article and article OR name runs

3.4 Entrypoint Tasks

We submitted some basic runs for both the AllInContext and the BestInContext tasks. These submissions started from top 1500 focused runs without overlap.

For AllInContext we grouped the retrieved elements by article and scored the articles by aggregating the element scores (for top 1500 elements only). We experimented with both maximum and average as aggregation functions. Within each article, we simply kept all elements that made it to the top 1500 of the original focused run. Thus, our AllInContext runs are nothing more than a re-ordering of the Focused run.

For BestInContext we did something similar. Again, we group by article, but now we order articles by the sum of the element scores, since we want articles with a lot of good material on top. Our assumption is a user wants to see all relevant passages in an article, thus as a best entry point, we simply returned the first element in document order that was in the focused top 1500. We did not extend our best entry point run with articles that did not made it to the focused top 1500, thus our BestInContext runs typically contain fewer than the allowed 1500 entry-points.

The approach was applied to both the plain and article weighted elements runs. Figure 5 shows our results in the AllinContext task. The two element runs (ARTorNAME_STAR_AVG and ARTorNAME_STAR_MAX) are indistinguishable. The article run (ARTorNAME) is clearly better. Apparently, in this

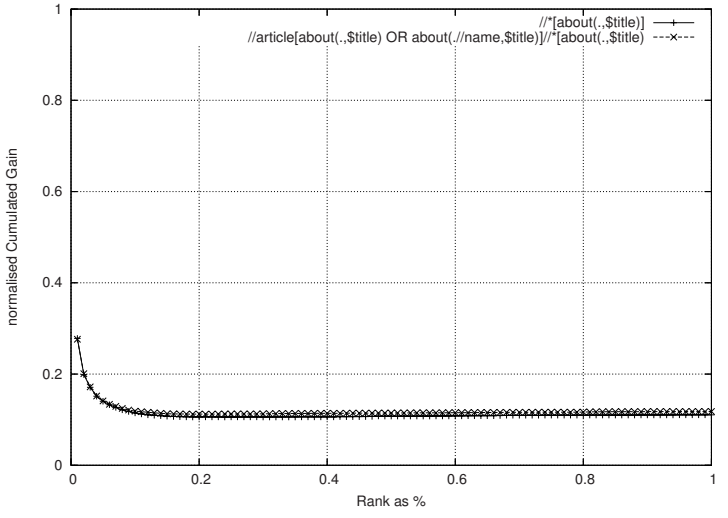

Fig. 4. Normalised Cumulative Gain for //* runs with and without article weighting

collection, selecting the best elements inside an article is still a hard task, and the best strategy is to simply return the entire article. Similar results were found on the BestInContext task:

4 Multimedia Track

The CWI/Utwente participation in the multimedia track is without any image processing; all our submitted runs are purely text based. Still, we adapted our runs to cater for the multimedia information needs. We augmented the Wikipedia collection with metadata, we filtered the results for images and we added some knowledge about the example images to our queries. Below these approaches are discussed in detail.

4.1 Augmenting the Collection with Image Metadata

The MMfragments is similar to the Ad Hoc track in that it asks for fragments from the Wikipedia collection. The main difference is that the information needs in this task have a clear multimedia character. The idea is that a system would return fragments containing relevant images together with relevant text. The PF/Tijah system and the Language Models used are designed for returning relevant (structured) text. To be able to work with images, we tried to get extra textual information in, to help us decide which are the relevant images. We did this by adding text from the image metadata document as available in the Multimedia images (MMimages) collection. Each $< image >$ tag in the collection is augmented with the corresponding metadata from the MMimages collection. We did not try to separate names, users and captions, we simply added the contents of the entire metadata document as text under the image tag.

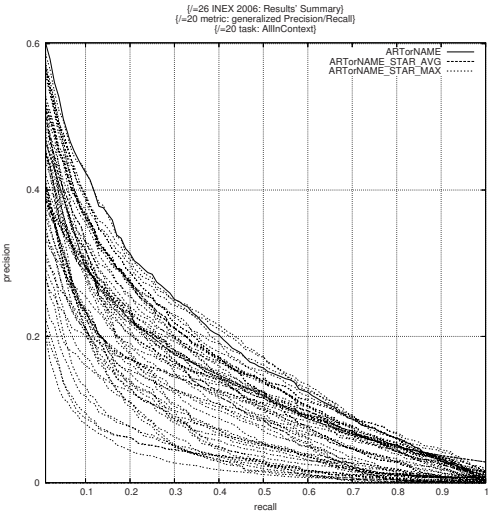

Fig. 5. AllinContext, generalised precision/recall: cwi/utwente compared to the competition

4.2 Filtering Results

Since the MMfragments deals with multimedia information needs, it seems wise to return only fragments that contain images. We made sure this was the case by filtering our results. Not all $< image >$ tags in the Wikipedia correspond to images that are actually part of the INEX multimedia collections; images that are not part of these collections will not be visible to users during assessments. Therefore, we also removed all results that contained references to images that are not in the collection. This way, we made sure all our returned fragments contain at least one *visible* image from the multimedia collections.

4.3 Experiments

We participated in the MMfragments and MMimages tasks. For both tasks, we experimented with relatively simple text only queries, aiming to show that text only queries can be competitive. Below we discuss our experimental results.

MMfragments. For MMfragments we submitted one full article run and three element runs. For the element runs, we did not directly use the given *castitle*, but we experimented with runs of the form: `//*[about(.,X)]`, where X contained some set of terms taken from the topic. We experimented with the following sets of terms:

STAR_TITLE the *title* field

CAS_noMM the terms from the *castitle* field without the visual examples and concepts

CAS_MMtext the terms from the *castitle* field plus the textual terms from the example images metadata documents (again no concepts).

The article run (*ART_TITLE*), was based on the title field of the topic only.

These queries were run against the augmented collection and the results were filtered to make sure all returned fragments contain images from the collection. The results are very disappointing; with mean average effort precision values of around 0.001. The main reason for this is that, like in the Ad Hoc task, we remove overlap from the results. This means we submitted Focused runs, while the results are evaluated using the thorough setting. The results for our thorough runs that still contain overlap show median performance, see Figure 6.

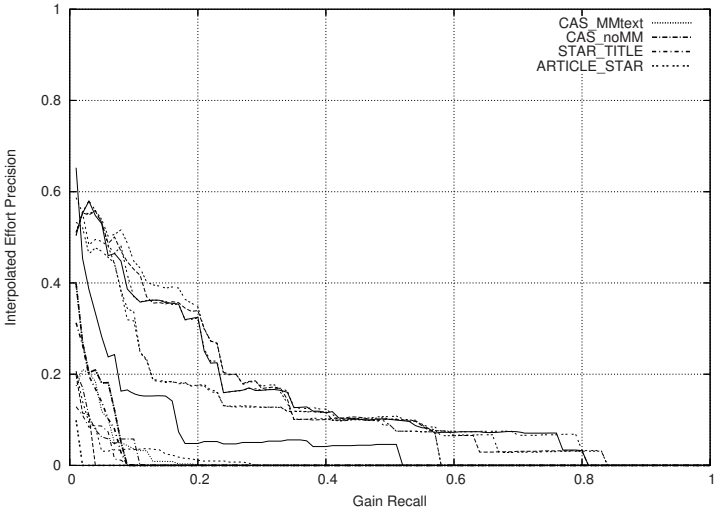

Fig. 6. MMfragments results: cwi/utwente runs compared to competition

MMimages For MMimages, we submitted two very basic runs:

article-title An article run, only using the title field of the topic:
`article[about(.,$title)]`

cas-noMM A *castitle* run ignoring the visual hints, we removed all image examples and concept references from the NEXI query.

Figure 7 shows these basic runs give highly competitive performance on the MMimages task. Clearly it is hard to beat a text only baseline on this task.

5 Conclusion

PF/Tijah was used at INEX for the first time. The flexibility of the system and the ease with which it allows the modification and combination of XML data

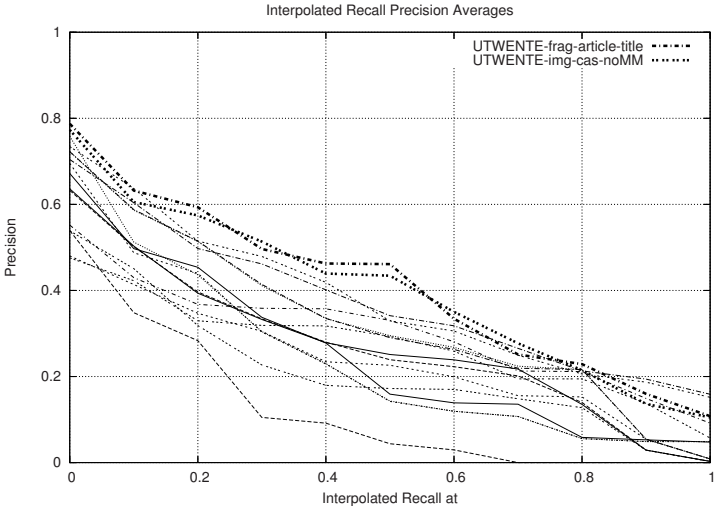

Fig. 7. MMimages results: cwi/utwente runs compared to the competition

makes it a very useful tool for (XML) IR experiments. For short article only queries, the system is reasonably fast; for larger queries it takes a bit longer, but for IR experiments, this is still acceptable.

Without adapting the models we used for the IEEE collection in previous years, the unigram language modelling approach retrieves elements that are often (almost) complete articles or links to them. Moreover, for the AllInContext and BestInContext tasks, the tasks most close to a realistic setting and to the way the assessment procedure is set-up, retrieving complete articles rather than smaller elements appears to be a useful strategy. Still, among the known relevant elements are many smaller elements like paragraphs and sections. Perhaps these smaller relevant elements appear in clusters. Further study is needed to investigate whether this explains the success of the article retrieval strategies.

Our text only approach to multimedia retrieval was very successful on the MMimages task, but less so on the MMfragments task. Perhaps a smarter way of filtering the results is needed to retrieve the appropriate multimedia fragments.

References

1. Arvola, P., Junkkari, M., Kekäläinen, J.: Generalized contextualization method for xml information retrieval. In: CIKM '05. Proceedings of the 14th ACM international conference on Information and knowledge management, New York, NY, pp. 20–27. ACM Press, New York, NY (2005)
2. Boncz, P., Grust, T., van Keulen, M., Manegold, S., Rittinger, J., Teubner, J.: Monetdb/xquery: a fast xquery processor powered by a relational engine. In: SIGMOD '06. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, New York, NY, pp. 479–490. ACM Press, New York, NY (2006)

3. Hiemstra, D., Rode, H., van Os, R., Flokstra, J.: Pftijah: text search in an XML databases system. In: Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR) (2006)
4. List, J., Mihajlovic, V., Ramirez, G., de Vries, A., Hiemstra, D., Blok, H.: Tijah: Embracing ir methods in xml database. Information Retrieval 8(4), 547–570 (2005)