Information Retrieval and Large Text Structured Corpora*

Fco. Mario Barcala[1], Miguel A. Molinero[2], and Eva Domínguez[1]

[1] Centro Ramón Piñeiro, Ctra. Santiago-Noia km. 3, A Barcia,
15896 Santiago de Compostela, Spain
barcala@freeresearch.org, edomin@cirp.es
[2] Depto. de Informática, Universidade de Vigo,
Campus As Lagoas, s/n, 32004 Ourense, Spain
molinero@uvigo.es

1 Introduction

Conventional Information Retrieval Systems (IRSs), also called text indexers, deal with plain text documents or ones with a very elementary structure. These kinds of system are able to solve queries in a very efficient way, but they cannot take into account tags which mark different sections, or at best this capability is very limited.

In contrast with this, nowadays, documents which are part of a corpus often have a rich structure. They are structured using XML (Extensible Markup Language) [1] or in some other format which can be converted to XML in a more or less simple way. So, building classical IRSs to work with these kinds of corpus will not benefit from this structure and results will not be improved.

In addition, several of these corpora are very large and include hundreds or thousands of documents which in turn include millions or hundreds of millions of words. Therefore, there is the need to build efficient and flexible IRSs which work with large structured corpora.

There are several examples of IRSs based on corpora [2] [3], of search methods over large corpora [4], and Chaudhri et al. [5] even introduce a review of different technologies that can be used to build generic IRSs based on XML. However, there are no comparative analyses or studies about technologies that can be used to build IRSs based on large structured corpora.

Since these IRSs can be wide ranging, in this work we will focus on those which work with corpora that do not include any morphosyntactic annotation and are structured in XML format. All topics studied in this paper will also be useful for annotated corpora (although the study need to be completed for the latter) or for corpora without XML format (because if corpora are correctly structured, they can be easily converted to XML format).

* Partially supported by Ministerio de Educación y Ciencia (MEC) and FEDER (TIN2004-07246-C02-01 and TIN2004-07246-C02-02), by MEC (HF2002-81), and by Xunta de Galicia (PGIDIT02PXIB30501PR, PGIDIT02SIN01E and PGIDIT03SIN30501PR).

R. Moreno Díaz et al. (Eds.): EUROCAST 2005, LNCS 3643, pp. 91–100, 2005.

So, first we will introduce the corpus used to illustrate the study. Next we will present the main alternatives for building IRSs based on large structured corpora. After that we will evaluate two technologies that would seem to be the flagships in this research field, previously defining the needs of that kind of system: on one hand Oracle[1] [6], a Relational Database Management System (RDBMS) that includes XML facilities, and on the other hand, Tamino[2] [7], a native XML indexer. Finally, we will show conclusions and the technology which best fits the established needs.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT document (document_header, document_content)>
<!ELEMENT document_header(identifier,medium,name,publishing_year,publisher)>
<!ELEMENT document_content (section+)>
<!ELEMENT section (name, new+)>
<!ELEMENT new (new_header, new_content)>
<!ELEMENT new_header (identifier, author+, theme+)>
<!ELEMENT new_content (headline?, abstract?, caption*, body)>
<!ELEMENT headline (paragraph+)>
<!ELEMENT abstract (paragraph+)>
<!ELEMENT caption (paragraph+)>
<!ELEMENT body (paragraph+, note*)>
<!ELEMENT paragraph (sentence+)>
<!ATTLIST paragraph distinct (other_language) #IMPLIED>
<!ELEMENT sentence (#PCDATA|note_reference|distinct)+>
<!ATTLIST sentence distinct (other_language) #IMPLIED>
<!ELEMENT note (paragraph+)>
<!ATTLIST note identifier ID #IMPLIED
<!ELEMENT note_reference EMPTY>
<!ATTLIST note_reference reference IDREF #REQUIRED>
```

Fig. 1. Newspaper DTD. Elements not defined are of type #PCDATA.

2 Definition of the Target Corpus

In this work we show technologies, techniques and methods to build IRSs based on large structured corpora which will be illustrated over a real corpus used as example, the CORGA: "Reference Corpus from Present-day Galician"[3]. This, which in its XML version has more than eight million words distributed in hundreds of documents, will also be used to evaluate the technologies studied.

CORGA documents can be newspapers, magazines or books, so we have a DTD (Document Type Definition) [1] for each kind of document. Figures 1 and 2 show the DTDs for newspapers and books respectively[4].

[1] Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

[2] Tamino is a Software AG product.

[3] Freely available at http://corpus.cirp.es/corga

[4] Actually, the DTDs used in CORGA are more complex. Here we only show a simplification of them to increase the clarity of explanations.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT document (document_header, document_content)>
<!ELEMENT document_header (identifier, medium, title, author+,
          publishing_year, publisher, theme+)>
<!ELEMENT document_content (preface*, dedication*, cite*, caption*,
          body, appendix*)>
<!ELEMENT preface (header?, content)>
<!ELEMENT header (author+, theme*)>
<!ELEMENT content (head?, caption*, body)>
<!ELEMENT body (part+, note*)>
<!ELEMENT appendix (header?, content)>
<!ELEMENT part (head, dedication*, cite*, caption*, part_body)>
<!ATTLIST part type (chapter|part) #IMPLIED>
<!ELEMENT head (paragraph+)>
<!ELEMENT part_body (paragraph+ | part+)>
<!ELEMENT dedication (paragraph+)>
<!ELEMENT cite (paragraph+)>
```

Fig. 2. Book DTD. Elements not defined correspond to those for newspaper DTD (see figure 1).

3 Building Alternatives

To build IRSs based on corpora several aspects have to be taken into account:

- It is very important to separate the corpus document structure from that of the IRS, that is, corpus document structure cannot condition the IRS one and vice versa, and we have to think of two different systems. In this way, we avoid penalising either the expressiveness of corpus DTDs or system performance.
- With large corpora, a common structure of documents for the IRS has to be designed. Typically there are different kinds of document in the corpus (newspapers, magazines, books, etc.). So, if this variability is maintained in the system, more queries (or more complex ones) have to be made to solve users' queries, and performance will be penalised.
- In these systems the priority is speed in retrieval. Normally these systems about corpora are updated once every three or six months (or never if the corpus is closed), so, in general, it is not important if system updates take several hours or days.

Nowadays two research lines are being followed to build IRSs which work with XML documents. The first one is based on the adaptation of XML documents to the relational model, which are then inserted into a RDBMS. The another one is to introduce XML documents directly into an XML-native or XML-enabled Management System, which allows the documents to be worked on the original XML format.

3.1 Relational Database Management System

There are several ways to represent XML documents in an entity-relation model [5], and in full in a relational model: generic automatic approaches, which allow any kind of XML document to be introduced into a database, or ad-hoc manual ones, which allow only some kinds of XML document to be introduced into it.

Although the second approach force us to define ad-hoc structures and procedures used to introduce our kinds of document into the database, for large corpora it is better to choose this alternative, because it allows us to obtain a higher performance in the retrieval process.

There are several ways to define this ad-hoc structure, and we have to define one which minimises relations in order to improve performance as much as possible. Figure 3 shows the entity-relation model chosen to test this technology.

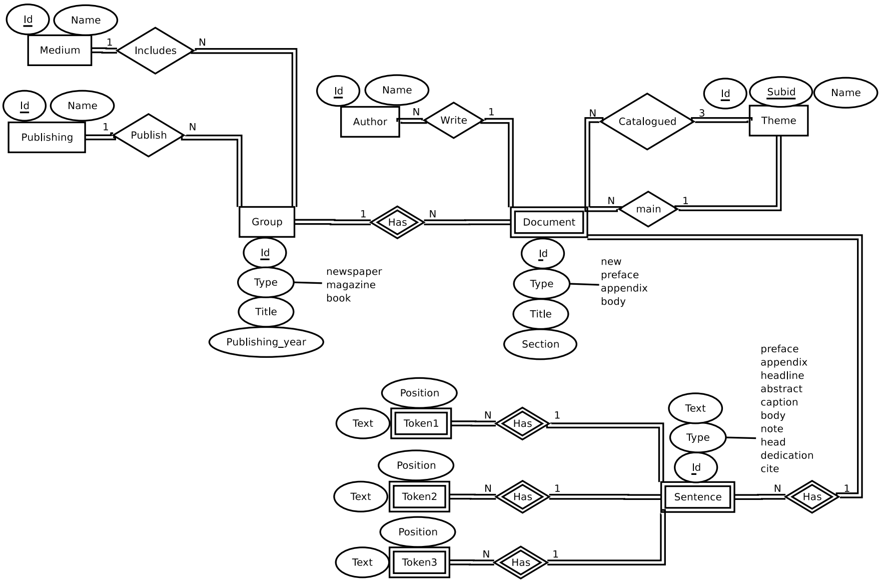

Fig. 3. Entity-relational model

3.2 Native XML Manager

In the case of a native XML manager, it is also necessary to define a common DTD which includes all kinds of document of the corpus. This will avoid a loss of performance associated with the query of different kinds of document, as we have mentioned earlier.

This common structure must also be simple and homogeneous, and have few hierarchies in order to obtain the highest performance, since queries will include fewer structural elements. Figure 4 shows the common XML format DTD chosen to test this technology.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT group (group_header, document+)>
<!ELEMENT group_header (title, type, publishing_year, publisher,
          author*, theme*, main_theme)>
<!ELEMENT document (author+, theme+, main_theme, sentence+)>
<!ATTLIST document type (New|Preface|Appendix|Body) #REQUIRED
           section CDATA #IMPLIED
                   title CDATA #IMPLIED>
<!ELEMENT sentence ((#PCDATA|reference_note|distinct)*, tokens)>
<!ATTLIST sentence type (Preface|Appendix|Headline|Abstract|Caption|
                         Body|Note|Head|Dedication|Cite) #REQUIRED>
<!ELEMENT tokens (token1+, token2*, token3*)>
<!ELEMENT token1 (#PCDATA)>
<!ATTLIST token1 pos CDATA #REQUIRED>
<!-- Ditto for token2 and token3-->
<!ELEMENT distinct (#PCDATA|note_reference)*>
<!ELEMENT theme (id, subid)>
<!ELEMENT main_theme (id, subid)>
```

Fig. 4. Common DTD

4 Evaluation

We have tested the main product for each the two technologies mentioned: Oracle (9ir2 version), a widespread RDBMS, and Tamino (4.2.1 version), a native XML manager with many capabilities. Two main criteria were used in the evaluation:

- **Flexibility:** Taking into account the requirements of IRSs based on corpora, we determine how deeply each technology verifies them.
- **Performance:** Using a representative set of queries, we test the performance of the system running them and measure the time needed to obtain the results.
  Despite the fact that there are some benchmarks for XML IRSs [5], they are too generic and not oriented to the needs of systems based on corpora. Furthermore, these benchmarks are not valid for both kinds of technologies, thus there are benchmarks for RDBMSs and different ones for XML native indexers, but there are no benchmarks to measure the performance on both technologies at the same time. Therefore, we have designed a representative set of queries to make these tests.

4.1 Flexibility

IRSs based on corpora can have a heterogeneous range of requirements, and types of query can be very different from one system to another, but there are some generic needs that most systems should have:

1. **Statistical capabilities:** Capabilities to obtain numerical values at different levels. For example, to count the number of cases and documents which match a query.

2. **Additional information:** Capabilities to offer supplementary information with the returned results. For instance, to return sentences which verify search criteria but also showing the author of the relevant document, publisher name, etc., that is, additional data taken from the corpus document structure.
3. **Match highlighting:** To highlight the term or terms which produced the matching.
4. **Context:** To show not only the matching cases but their context as well. For instance, in the CORGA example it must be possible to show the whole sentence where the match was found, or *n* words before and after the match, or even *n* sentences before and after the sentence with the match.
5. **Index/Search methods:**
   (a) Exact matching queries.
   (b) Diacritical mark sensitive or insensitive queries.
   (c) Case sensitive or insensitive queries.
   (d) Boolean queries.
   (e) Proximity queries. That is, queries about words separated by less than a given distance.
   (f) Several tokenization and indexation criteria in the same database. For example, using the same database, a user could decide to make a query that is sensitive or non-sensitive to accents.
   (g) Excluding marked text from index. For example, in the CORGA example, exclude from indexation all text within the *distinct* element.
   (h) Ignoring certain characters on indexation. Sometimes corpora builders use in-line marks that we want to see in the results, but we do not want to index them. For instance, the use of brackets to mark recomposed text from a medieval edition.
6. **Charsets:** To allow several of the most commonly used charsets.
7. **Use of wildcards:** To allow the use of wildcards or regular expressions in queries.
8. **Results browsing and navigation:** It must be possible to navigate between pages of results without having to repeat the query.
9. **Ordering:** To order the results using one or several criteria at a time.
10. **Structural relationships:** The query language should be flexible enough to build a wide range of queries.

Evaluation

In figure 5 we show the evaluation of Oracle-SQL-Relational and two query languages of Tamino XML Native database: XQuery, a working draft of World Wide Web Consortium [1] and X-Query, a Tamino proprietary language. The following conclusions can be extracted:

- X-Query language has the lowest flexibility, so we will take into account only the XQuery alternative in the following discussions.

Requirement	**Oracle**	**Tamino XQuery**	**Tamino X-Query**
1	several at a time (tokens needed)	one at a time (tokens needed)	**No**
2	Yes	Yes	limited
3	Yes	Yes	**No**
4	Yes (several possibilities)	Yes	**No**
5a	Yes	Yes	Yes
5b	Yes	Yes	Yes
5c	Yes	**No**	**No**
5d	Yes	Yes	**No**
5e	NEAR	PROXIMITY-CONTAINS	ADJ,NEAR
5f	Yes	**No**	**No**
5g	Yes	Yes	Yes
5h	SKIPJOINS	**No**	**No**
6	Yes	Yes	Yes
7	%,_	*,?	*,?
8	Yes	Yes	Yes
9	Yes (several criteria)	Yes (several criteria)	Yes (several criteria)
10	SQL (high flexibility)	XQuery (very high)	X-Query (low)

Fig. 5. Flexibility evaluation. Numbers in the first column correspond to the requirements listed in section 4.1.

- Both Tamino and Oracle can obtain statistical information about any level of documents, whenever those levels are included in the structure of the representation of documents.

  This requirement can be problematic, because it could force the replication of data. For example, in the entity-relation model in figure 3, a document is broken down until word (token) level. *Token1*, *token2* and *token3* contain sequences of one, two and three words from each sentence (the structure is needed to count the number of cases that match a query). So, the text is replicated in sentence and token structures.

  Moreover, only Oracle allows us to obtain different statistical information by sharing calculations, thus considerably improving system performance.
- Oracle offers more possibilities to show the context of searched terms.
- Both technologies have proximity operators, but Tamino does not allow us to build case-sensitive indexes, or define different criteria for tokenization and indexation for the same database, or ignore certain characters in the indexing process.
- Both Tamino and Oracle allow the use of different charsets, including UTF-8.
- Both technologies allow us to order the results by several criteria at a time.
- XQuery is even more flexible than SQL, since it has a more complex syntax that allows more complex structures to be represented and manipulated.

Although it would appear that Tamino has almost all the required needs, we must take into account that the proposed needs are minimal for several IRS based on corpora, and the absence of one of them could make all the difference between a useful or useless system.

Moreover, XQuery is still a working draft and is not yet completely implemented. Oracle would therefore seem to be the best alternative, from the flexibility point of view, for building these kinds of system.

4.2 Performance

To evaluate performance, first we define a representative query set, covering the topics explained in the flexibility section, and then we show the elapsed time needed for each technology to solve the queries. These queries are based on the studied corpus (CORGA), but the topics covered are generic enough to be applied to any other IRS based on structured text corpora.

Query set

Queries are defined as follows:

- **Q1:** To obtain the number of documents and cases from sentences which contain the expression "sen embargo" for each type of medium, main theme and lustrum ("sen embargo" means however in the Galician language).
  The system includes three kinds of medium, six different main themes and six lustrum, so thirty numbers must be obtained. We measure the time elapsed between starting the query and checking all results. It is a measure of efficiency in sharing calculations.
- **Q2:** To obtain the sentences which contain the expression "sen embargo" with their document title, publisher, authors, medium, themes, publishing year, and type of sentence ordered by publishing year, medium and main theme. Match words are returned highlighted.
  We measure the time elapsed between starting the query and showing all required values. It is a computational cost measure about showing all associated elements and highlight and sort operations.
- **Q3a:** To obtain the number of documents and cases from sentences which contain a word with "pre" prefix.
- **Q3b:** To obtain the number of documents and cases from sentences which contain a word with "ado" suffix (the "ado" termination in Galician is usually used for participles of verbs).
- **Q3c:** To obtain the number of documents and cases from sentences which contain a word with "poñ" infix (in Galician "poñer" means to put, "repoñer" means to put back, and "pospoñer" means to postpone).
  Q3 queries are different measures of text index efficiency.
- **Q4:** To obtain cases from sentences which contain a word with "in" or "im" prefix, but only in news headlines from newspapers and with a context of fifteen words. As neither technology supports built-in word context, in this

case the queries return all the necessary information to perform this operation at application level.

We combine a structural complex query with word context retrieval to measure its computational cost.

Evaluation

We have tested these queries on a PC Intel Celeron at 2.4GHz, with 512 MB of memory and Windows XP Professional over a CORGA six hundred thousand word subcorpus (uniformly distributed between newspapers, magazines and books) with previously shown structures.

Although this hardware is not the most suitable for testing these kinds of system, which need powerful servers to manage millions of words, it provides us with conclusive results about the performance difference between both technologies tested.

As we can see in figure 6, the most important difference is in Q1 query, due to the inability of Tamino to calculate several statistic values by sharing calculations. So, once again Oracle leads the results of the tests, confirming it as the best option for building this kind of system.

Query	**Tamino XQuery(4.2.1)**	**Oracle (9ir2)**
Q1	585.203s	10.891s
Q2	33.922s	14.531s
Q3a	106.719s	5.828s
Q3b	118.266s	34.172s
Q3c	98.750s	3.687s
Q4	26.545s	11.187s

Fig. 6. Performance evaluation. It shows time duration of proposed queries in seconds.

5 Conclusions

First, it is necessary to emphasise that it is mandatory to transform documents of the corpora into a common format when managing large amounts of information. This will allow us to query all documents using a unique query and to improve the performance of the system. By doing so we will avoid problems with performance and result management.

Furthermore, nowadays, the technologies used to build IRSs are not prepared to satisfy corpora users' requirements. So, in the near future the development of new add-ons which take them into account is needed. There are some timid attempts to include basic linguistic operations (sensitivity to accents, umlauts, etc., theme searches, etc.) based on localization, but it is time to incorporate syntactic techniques [8] [9] into commercial systems to enable the building of more versatile IRSs based on corpora.

Furthermore, traditional technologies manage updatable systems, but companies need to develop more optimisation options for systems which give very high priority to query performance regardless of the very high penalty in updates.

Finally, XML technologies are being developed very rapidly, but they still have to settle. The speed of this evolution prevents robustness and clear development of systems, making it difficult to put them in production.

Against this, RDBMSs have a high robustness, flexibility and performance, due to their long life in the market. Taking into account our test results, the fact that these systems include text index capabilities and the drawbacks of other technologies, at the moment we propose Oracle as the first option for building IRSs based on large structured corpora.

References

1. XML. In *http://www.w3c.org*, 2/5/2005.
2. María Sol López Martínez. CORGA (Corpus de Referencia del Gallego Actual). In *Proc. of Hizkuntza-corpusak. Oraina eta feroa*, pages 500–504, Borovets, Bulgaria, Sept. 2003.
3. Mark Davies. Un corpus anotado de 100.000.000 palabras del español histórico y moderno. In *Proceedings of Sociedad Española para el Procesamiento del Lenguaje Natural* pages 21–27, Valladolid, Spain, 2002.
4. Mark Davies. Relational n-gram databases as a basis for unlimited annotation on large corpora. In *Proceedings from the Workshop on Shallow Processing of Large Corpora*, Lancaster, England, pages 23–33, March 2003.
5. Akmal B. Chaudhri, Awais Rashid and Roberto Zicari. XML Data Management, Native XML and XML-Enabled Database Systems, *Addison-Wesley*, March 2003.
6. Oracle. In *http://www.oracle.com*, 2/5/2005.
7. Tamino. In *http://www.softwareag.com*, 2/5/2005.
8. Jesús Vilares, Miguel A. Alonso and Manuel Vilares. Morphological and syntactic processing for Text Retrieval. In *Database and Expert Systems Applications, volume 3180 of Lecture Notes in Computer Science, Springer-Verlag*, Berlin-Heidelberg-New York, pages 371–380, 2004.
9. Miguel A. Alonso, Jesús Vilares, and Víctor M. Darriba. On the Usefulness of Extracting Syntactic Dependencies for Text Indexing. In *Artificial Intelligence and Cognitive Science, volume 2464 of Lecture Notes in Artificial Intelligence, Springer-Verlag*, Berlin-Heidelberg-New York, pages 3–11, 2002.