Large-scale structured linguistic corpora: Methodology and information retrieval systems (summary)

Fco. Mario Barcala Rodríguez
University of A Coruña

1 Introduction

This work deals with a particular case of information retrieval (IR) systems that use structured information: that of linguistic systems that work with large amounts of data (corpora), which places this doctoral thesis within the field of computational linguistics and, more specifically, within that of corpus linguistics. Although this field offers a wide range of possibilities, we choose to focus on those systems in which the information required by users, who will usually be linguists, is related to the frequency with which words occur or to the visualisation of examples in the context in which they were used.

These systems evolved practically alongside the development of computers. From the first monolithic search tools that used document collections that are now considered to be small-scale, the ever-increasing capacity of computers has enabled this evolution to take place, giving us our present-day internet-based query systems that work with large-scale corpora[1]. Our focus will be on these, analysing the different possibilities and technologies currently available for their development. But at the same time we will also put forward a generic methodology proposal for the creation of corpora[2], which are the data support for these IR systems.

We therefore offer an overall vision that covers both the construction and the subsequent exploitation of corpora, continually bearing in mind the use of state-of-the-art standards. As a result, we hope that on the one hand this work will be used as a guide for future projects, and on the other that it will guarantee the appropriate evolution over time of the proposals we put forward.

This summary is organized as follows. Section 2 describes the objectives of the doctoral thesis, Section 3 proposes a methodology for corpus construction, Section 4 studies its application to a specific case, that of the Corpus de Referencia do Galego Actual (Reference Corpus of Present-day Galician Language - CORGA) [21, 55], Section 5 makes general reference to the requirements common to the type of IR systems studied, in Section 6 we evaluate different technologies available for their development, in Section 7 we explain the IR system developed for the CORGA, and finally in Section 8 we present the conclusions reached as a result of the work done and future work.

[1]A large-scale corpus in this field is currently considered to be one that includes several tens or even hundreds of million words.

[2]The absence of published information in this regard was the reason for undertaking this challenge, which turned out to be one of the fundamental lines of our research.

2 Objectives

The starting point for this work can be considered to be the development of various natural language processing (PLN) tools within the framework of a collaborative agreement between the Centro Ramón Piñeiro para a Investigación en Humanidades (Ramón Piñeiro Centre for Research in the Humanities) [20] and the COLE (Compiladores y Lenguajes - Compilers and Languages) group [46] from the University of A Coruña.

After analysing the needs of the said research centre with regard to the CORGA project, the need for a dual restructuring became apparent: on the one hand the structure for existing documents had to be enriched, whilst on the other, it was necessary to develop a new query system that could make use of this new type of structure.

The present work arose both from the research carried out to meet these two needs and from our own interest in generalising the results so that they could be used in other contexts. Its objectives are:

- To define a methodological proposal for the creation of large-scale structured corpora.
- To define the requirements demanded of an IR system used to query a corpus.
- To evaluate the technologies currently available for developing systems of this kind.
- To propose an architecture for such a system.
- To apply the above proposals to a specific case, namely that of the CORGA project.

3 Methodology

The construction of a corpus is not such an easy task as it may initially seem. It is often the case that after the document structure has been defined and documents have already been added to the corpus, we are faced with the problem of a new text that we wish to add, but which does not fit the predetermined structure. When this occurs during the initial stages of the corpus creation it is not an excessively difficult task to modify and debug it so as to represent the new elements that had not been taken into account at the outset. However, when this occurs at a later stage, it becomes much harder to undertake the necessary changes, and the task may consume an unacceptable level of resources.

In this section we propose a sequence of phases that need to be introduced in the construction of extensible markup language (XML) [99] based text corpora in the field of corpus linguistics or lexicography, which ranges from the definition of the working methodology to the development of the corpus itself. There are numerous studies that explain the procedures to be followed when developing a given corpus [13, 18, 77], and others in which an attempt is made to define some general criteria that should be taken into account for corpus design and development [9, 15, 64, 78], but we have been unable to find works that generalise and organise both general and specific concepts in order to provide a full guide for different types of project.

The principles dealt with below describe the most important aspects that need to be taken into account, in the context of computational linguistics, when developing almost all kinds of monolingual structured text corpus that do not include any morpho-syntactic information whatsoever. We will therefore refrain to mention tasks and concepts that have to do with this latter type of information.

Figure 1 shows the five main stages needed to develop a corpus of these characteristics, which we will analyse in detail below. Although the proposed phases follow a chronological order, it will at times be necessary to return to a prior phase to refine or modify a previous decision. This return to an earlier phase or phases may happen either because it is convenient to organise the work in different parts, or because we detect an error that needs to be corrected. The less this latter circumstance occurs, the less the human and financial resources we will need to consume.

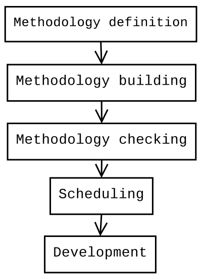


Figure 1: The stages in the construction of a corpus

On a transverse level, it is extremely important to record all the information relating to each of the phases proposed above, which should be reflected in one or more documents. The construction of a corpus commonly takes place over a substantial period of time, and if decisions are not recorded when they are taken they will be forgotten, leading to a high risk of inconsistencies appearing at a later stage.

3.1 Methodology definition

In methodology definition phase all factor which affect the corpus construction must be set. We define them as follows:

The objectives of the corpus

The objectives that lie behind the decision to develop a corpus are the first element that needs to be taken into account when defining the methodology. There may be various different reasons for constructing a corpus, but they can usually be placed within the overall objective of serving as a basis for a tool that will either allow users to undertake different types of philological study[3], or to enable tasks requiring its use[4].

A team sets out to develop a corpus because it has previously thought about the objectives or needs that the latter will be able to meet. Thus, its development should not be seen as a final objective in itself, but rather as a means through which real objectives, i.e. philological studies, machine translation of texts, etc. can be met.

These objectives will be decisive when it comes to define other elements of the methodology, since the type of corpus or the way in which the documents are structured will vary according to the kind of study to be undertaken.

When attempting to identify specific objectives it is obviously helpful to think about the use to which the corpus will be put and the possibilities offered by existing tools or by a possible IR system to be built ad hoc, but particular care should be taken to restrict the scope of our aspirations. We may initially come up with a host of specific objectives, but these will need to be pruned and prioritised to prevent the possibility of failure when it comes to perform the tasks required for its development.

[3]To consult a specific case of occurence, to prove or reject a theory, to look for examples that support or refute a thesis, etc.

[4]For example, tasks relating to machine translation or spelling correction.

Type of corpus

The second decision consists of determining what kind of corpus we are going to develop. Depending on the criteria employed, we can identify a multitude of classifications that define different types of corpus [9, 31]. Taking these studies as our starting point, together with other collections [13, 66], we here define the types of text corpus that we consider to be the most relevant in terms of their influence on how the methodology is defined:

- Reference corpus: a representative sample of the main varieties of a language.
- Specialised corpus: representative of a specific linguistic variety or specialised language.
- Monitor corpus: a corpus of constant size in which new material is included as older material is simultaneously removed.
- Fragment corpus: a corpus consisting of fragments of text.
- Whole text corpus: a corpus consisting of complete texts.
- Bilingual or multilingual corpus: a corpus containing texts in two or more languages.

Types of document

Before defining the structure of our documents it is essential that we decide what kind of document we are going to include in the corpus. We will therefore need to take into account the genre (literary, journalistic, etc.), the thematic areas, the medium of publication (journal, book or newspaper) and other specifically relevant criteria for the document set.

Documentary sources

We will need to draw up an initial list of the sources from which we are going to obtain our documents. We will have to check that they can in fact be obtained through the channels we are going to use, and that we will at all times be respecting copyright law when we process them and include them in our corpus.

This is also the appropriate moment to obtain some of the documents that we intend to include in the corpus. We will need a minimum of one or two of each kind, which we can use as models to guide us during the process of defining their structure.

Representation standards

There are two main trends in the development of corpora using XML:

- To define an ad hoc XML for representing documents.
- To use an available standard such as XCES [92] or TEI [89].

The decision as to whether to adopt one approach or the other depends on several factors. As a rule it is more useful to adopt a defined standard, since the fact that it will be used by many different groups gives it certain advantages, e.g.:

1. A greater compatibility between corpora, since it facilitates the exchange of information between different projects.

2. Improved training for project staff, minimising the impact of its cost when a person transfers from one project to another and favouring mobility between projects. Similarly, it also favours the introduction of training courses independent from the different projects, in which these standards can be explained.

3. Support optimisation of corpus tools. The existence of a common standard allows the development of different tools that can be used in various different projects.

However, there are some cases where the development of an in-house document structure may be advantageous:

- If we come to the conclusion that it is preferable for purpose of the project that the XML tags should be in a language other than English, e.g. the working language of the people involved in the project.

- If the standard does not allow us to refine the structure according to our needs, and we want to avoid having more tags than strictly necessary.

Nevertheless, even if we do decide to adopt the second approach, it is still useful to study and analyse the document structure proposed by the existing standards to help us best define our own structure. These standards have been created by experts, which is normally a guarantee that the decisions taken in them are the right ones.

We will henceforth assume in our explanations that we have opted to define our own XML, this being a more complex process. All the explanations will also be valid if the XCES standard is used, and we will point out any exceptions to this as we go along.

Document structure

Once we have determined our objectives, the type of corpus, the types of document to be include and the representation technology we are going to adopt, and assuming that we also have some text samples to be included in the corpus, we will need to define their structure. At first it is a good idea to think of all possible future uses of the corpus, since at this stage we should not forget to take into account any element that may be needed for processing in the future. However, we should not attempt from the outset to deal with absolutely every possibility and detail of tools that are going to be implemented in a far-distant future. Generally speaking, trying to cover absolutely everything from the very beginning means that the process of development will be too slow, and no significant progress will be appreciable for a long time. Nevertheless, in order to enable future changes to be made without incurring excessive costs, these aspects should be taken into account while the document structure is being defined, so that it can be shaped appropriately. In the majority of cases designers come up with a large number of different document structures. It is never easy to determine which one is the most appropriate, but it may be helpful to bear the following points in mind:

- **The physical appearance of the document**: since documents are often originally printed on paper, this appearance may lead us to failing to structure them appropriately.

  All too often, in an initial analysis of what form the document structure should take, we reach the conclusion that documents should be organised in pages, which in turn will be organised in lines. Although this type of structure may at times be perfectly justifiable, e.g. because of the importance of information about the exact text location in the publication, as a rule it offers more drawbacks than advantages during the development of the corpus.

  A good compromise solution will take into account all the various ways in which documents can be represented, as well as the different techniques for putting these into practice. For example, when using XML technology, as is our case, there are several different techniques enabling multiple hierarchies to be represented [30, 89].

- **The IR system**: in many cases the objective of the corpus we are building is to develop an IR system that will enable searches to be performed on the former. As we have indicated in a previous work [11], the document structure for this system may not only differ from that considered the most appropriate for the corpus itself, but on many occasions it is even preferable for it to do so.

  Thus, even if it is a good idea to take into account the document structure that we intend to incorporate into the IR system, we should not allow it to unduly influence our decision. If we do so, we would tend to oversimplify the documents and thereby lose out on expressiveness and flexibility.

Since the internal organisation of documents is crucial for the proper development of the corpus, we must pay particular attention to avoid making any serious design errors at this point. If the structure is correctly defined from the outset, we will not need to refine it in the future, or if we do it will only be to address specific questions of relatively little importance, which will lead to a saving in the costs generated by setbacks of this nature. Although the use of the XCES standard to a certain extent facilitates the task, we also have to decide which of the elements that this standard defines are those that we are going to use for the different kinds of document. XCES defines structuring elements for a wide variety of texts, but we should only use those that we really need.

Processing stages

In this section we define the different stages through which documents have to pass, from the original format and medium until they are fully adapted to the electronic structure that we have defined for our corpus. For each medium, format and type of text we will need to define a protocol that provides a detailed description of each stage, whether automatic or manual, that the texts will go through before they form part of the corpus. The protocol will ensure that all the decisions taken are clearly expressed and will be useful in training new members for the team of linguists who will be working on the manual tasks.

It is often the case that with the passage of time some of the details of these stages get forgotten. The existence of a series of protocols will avoid having to repeat discussions about decisions that have already been made and therefore reduce the delays and inconsistencies in the project as a whole.

We identify six generic processing stages[5], which are shown in figure 2 and described below:

[5]One or more of these stages may need to be modified and/or adapted to meet the needs of a given corpus, or even omitted altogether. In this case, we propose a series of generic phases that can be adapted for use with different types of corpus and work procedures.

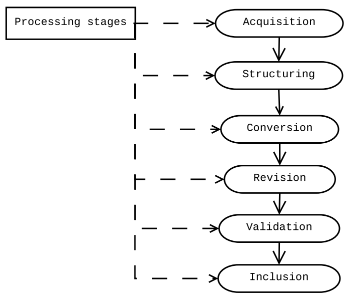


Figure 2: Processing stages

Acquisition

This is a manual stage that consists of acquiring an electronic version of the document to be processed. The difficulty of this task may vary enormously, mainly depending on whether the text in question is in printed or digital format.

Incorporating an on-line digital newspaper may be a relatively simple task: we would only have to download its pages. It would be even easier if the publisher agreed to provide us with an electronic version of the text(s) we are interested in. In this case we would only have to check that the digital document matchs exactly with the printed version that we want to process, but we would already have the electronic version itself.

It will be a much more time-consuming task, however, if the text to be incorporated is only available in hard copy. In this case we will have to scan the text page by page to convert the full content to our desired digital format. Thus, when the acquisition process has concluded, the document will consist of either an archive in the format of the scanning program used or a set of images containing all its pages.

Structuring

This is a manual stage that consists of structuring, in a common format that has to be defined and documented for each type of text, the digital documents that are the outcome of the acquisition stage. This format will facilitate the work to be done by the teams of computer experts and linguists in this and subsequent stages.

As a rule, the simpler this process is the greater will be the work required in subsequent stages, and vice-versa. The most important criterion when defining a structure is therefore to obtain a positive compromise between the work needed to create it in this stage and the work that will be involved in any subsequent processing.

Conversion

This consists of an automatic transformation of the documents from the structuring stage, with a view to adapting them to the previously defined XML format. It will therefore be necessary to create a script that will perform this conversion. The outcome of this stage will be an initial version of the document in XML.

Revision

Given the difficulty of the tasks that will have to be undertaken by the script in the preceding stage, and taking into account the compromise solution that should have been reached in the structuring stage for the intermediate format, we will have to suppose that the script will be unable to adequately represent all the necessary phenomena in the XML format defined for the corpus documents. So, they have to be revised and the protocols to be followed by the linguists must be written.

Validation

There are times when the tool used to perform the revision tasks does not allow us to undertake all the checks we consider necessary. It is therefore useful to define a validation stage, during which the documents that have already been revised are re-checked, either by using pre-existing tools or by ad hoc automatic scripts.

If a problem is detected at this stage, it will be necessary to return to the previous stage in order to solve it before running the document through the validation process again.

Inclusion

Once we have checked that a document fulfils all the necessary requirements to form part of the corpus, it can be included by simply putting it into the relevant directory that we have established for this purpose.

3.2 Methodology building

Once we have defined a working methodology, we will have to:

- Acquire and/or develop all the tools and resources needed to implement it.
- Install and configure these tools and prepare the working environment for the people involved.

This basically involves preparing everything we will need to develop the corpus: choosing DTDs [97] or XML schemas [100], obtaining the tools for the acquisition and revision stages, developing the scripts needed for conversion, obtaining the necessary financial and human resources and hardware, defining file storage policy, etc.

3.3 Methodology checking

Before starting the development of the corpus itself, we will need to perform exhaustive tests to ensure that the methodology is working correctly. They will only evolve certain members of the development team, preferably the coordinators of the linguistics and computer sections, respectively, who will revise all its constituent elements such as document acquisition programs, scripts, editors, etc.

The best method for performing this task is to take one or more documents of each type and pass them through the successive stages, from acquisition to inclusion. In this way we will be able to detect problems in the various processes during which the documents are transformed: scripts with bugs, problems with the operation and configuration of editing tools, etc. If an anomaly is picked up it should be corrected and the tests repeated until we are sure that all components are working correctly.

3.4 Scheduling

At this point we have to make a selection of texts from the document sources previously selected during the definition stage and schedule their processing over time. We will use the same criteria for planning their incorporation into the corpus as we used when defining the types of document to be included.

Perhaps the most effective procedure is to process the documents by type, to make training team members easier and reduce the probability of mistakes being made. However, there will be times when we need to obtain a large variety of text types in a short period of time, with only a few texts of each kind being processed, even though in this case the cost of training, and probably also processing quality, will fall short of the optimal.

3.5 Development

Finally, during the development stage we will apply the methodology we have defined to the different documents that were included for processing during the scheduling stage. At this point all the members of the development team will work on incorporating the texts into the corpus, and particular importance should be attached to holding regular meetings to unify criteria and look for a common solution to any doubts that may have arisen.

4 The CORGA corpus

In this section we will explain how we applied the above methodology in our particular case study: the CORGA corpus. In order to simplify our explanation we will refer to sections 3.1 and 3.2 jointly, that is, we will comment on the different points concerning methodology at the same time as we specify how we developed each part of it.

4.1 Methodology definition and building

The main objective pursued by the CORGA is to serve as a reference for studies, mainly of a lexical and morphological nature, on the Galician language in today's written media. Consequently, a crucial element for allowing access to the corpus is the development of some kind of tool that would be accessible to the user community and enable them to perform different types of search.

The CORGA fits perfectly in one of the types of corpus mentioned above, the reference corpus, since it is intended to constitute a representative sample of the main varieties of the written language. It is made up exclusively of texts written in Galician language and published from 1975 onwards, since that was the year when the first signs of standardisation started to appear in different areas. Given that the aim is to collect the linguistic richness or diversity as from that date, there are no restrictions with regard to genre, theme, type of publication or author.

With regard to document sources, the first ones where chosen on the basis of ease of access so as to get the project off the ground as soon as possible.

Document structure

After a detailed analysis of the XCES standard, we decided to define an XML of our own to represent the document structure. This enabled us to use tags in Galician and thus speed up the training of the linguists as they joined the project. We finally opted for six different types of document: newspapers, journals, books, collections of short stories or essays, plays and collections of plays.

We defined a separate DTD for each of these six categories that reflects their internal structure. Figures II.2.1 and II.2.2 of the thesis show some of the definitions common to all the DTDs, whilst figures II.2.3 to II.2.8 show those that correspond to each different type of document.

Decision-making criteria

As can be seen from the structure defined for the corpus texts, the minimum structuring unit that we take into account is the sentence. The reason for this decision is that although in the present work we focus on the CORGA as a non-tagged corpus, from the outset we have kept in mind the possibility of it being tagged, or even analysed as a treebank or parsed corpus.

On the other hand, in this case we made the decision to use DTDs instead of XML schemas because when we started to construct the corpus using the XML standard there were no sufficiently developed tools available that would allow us to work adequately with schemas.

Processing stages

We next describe the process of transformation undergone by the documents from the time when they become available to us until they are included in the corpus, a process which varies according to the type of text in question.

Acquisition

The simplest case of acquisition of a digital version of a document is that of PDF [1] documents, which are commonly available for downloading from web pages. For this purpose we just use a web browser and then download the document from its internet location. On the other hand, in the case of HTML [98] documents we have to use different strategies at different times during the downloading process. For certain documents consisting of various pages the use of tools such as wget [37] or HTTrack [74] can be useful, and if we are unable to download all the associated pages properly, then the only alternative is to download them one by one, using the web browser.

Printed documents, however, are an entirely different case, since they have to be digitalised by means of a scanner, which involves quite a lot more work. In our particular case we used Omnipage Pro 12[6] to help us perform this task. We defined different profiles within the software to deal automatically with the different forms in which we might receive the documents: using the automatic or manual page feed; single or double-sided pages; different commonly used page sizes; etc. This led us to the creation of various configuration files with different profiles that could be loaded in the program prior processing a document, which avoided having to manually specify every parameter on each occasion, thereby saving time and reducing the possibility of a mistake being made.

Each printed document processed is saved in a file with the Omnipage software format and stored to its appropriate place. Similarly, PDF files are saved in their own format, whilst for each HTML document a directory is created that contains the HTML pages constituting it.

Furthermore, in order to avoid any confusion between the use of the different tools used during this stage, we defined a detailed protocol for each step to be followed by the linguist responsible for acquiring a document. The existence of a set of guidelines not only saves time in training new staff, but also when it comes to settling frequently occurring doubts.

[6]Omnipage is a registered product of Nuance Communications, Inc.

Structuring

Once saved, the acquired documents have to be structured to adapt them to the intermediate form of representation that we have determined. In our case we use files and directories that will vary according to the type of document in question.

The computing tools used to carry out this task will depend on the origin of the document. Thus, if the document that we want to structure is composed of one or more HTML files, we will use a web browser and a text processor to create the desired format, using copy and paste functions.

If the original document was in hard copy format, and thus what we now have is the digital version in the Omnipage format, we will have to pass it through an OCR program, which in our case will also be the Omnipage Pro 12: the corresponding regions will be delimited in order to export them to text, which will be revised and the errors arising during the character recognition process corrected. In this regard, it is useful to draw up a list of typical errors which will enable the correction process to be partially automated. Structuring will finally be completed by using a text processor to adapt the corrected text to the chosen format.

If, however, the document to be structured is in PDF format, a similar process to that used for hard copy will have to be adopted. We will use Omnipage to export the text after the relevant regions have been defined. The only difference being that in this case there will be no need for OCR error correction, since these will not exist.

Conversion

Once the documents have been structured in the manner described in the previous section they will undergo a conversion process to generate an initial version of the corresponding XML document.

In our case we develop a series of transformation scripts[7] that take the directory structure defined for each document and reconstruct this XML version. Amongst the functions these scripts perform are the following:

- Create the main XML structure.
- Introduce the general heading information.
- Correctly structure the divisions for books or collections, and the sections for newspapers.
- In the case of newspapers and journals, insert the news items in the XML in the order established by the file name numbers.
- For collections, correctly structure and fill in the headings of the different elements.
- In the case of plays, correctly identify and fill in the characters.
- Segment the text into paragraphs, and these in turn into sentences. Given that only paragraphs are delimited during the structuring stage, using carriage return, we had to develop a sentence segmenter that would take into account abbreviations, acronyms, initials and other linguistic phenomena which make difficult the segmentation process.

[7]When we talk about scripts in the context of the CORGA project we refer to small programs developed basically with Perl [65, 93], XSLT [88, 103], libxml [40] and bash [33, 58].

Revision

During this stage the XML documents generated in the previous stage will be manually revised and corrected since, as is to be expected, the scripts will never leave the documents perfectly structured[8]. The following are some of the tasks that will have to be performed by the members of the team of linguists during this stage:

- Checking that the sentences have been correctly segmented. Although the sentence segmenter is reasonably sophisticated, it is not perfect and therefore errors have to be corrected manually.
- Placing photograph captions in their proper place: since these were not marked up in the structuring stage, they will need to be relocated in the document.
- Correcting errata: the documents may contain errata[9] that have reached this stage intact and therefore must be corrected.
- Inputting references for the footnotes: even though the footnotes are marked up and detected, this is not the case of their references, so they have to be inputted.

For these tasks we decided to use the XMLMind XML Editor program [67], due to its ease of handling and its availability as a free download. Although it allows an XML document to be processed without the need for an associated style page, the work is quite cumbersome: the document view is extremely homogenous since parts are not highlighted against each other, it is impossible to change the font size for viewing, the default font size is somewhat small for use with a normal screen resolution and changes to all the document attributes are performed in a separate, free-standing area. For this reason we decided to define a style page that would enable us to visualise and process the documents in a much simpler way.

Validation

Although some validation is done with the tool used for the revision process to avoid mistakes being committed, in our case we built a series of scripts responsible for validating the content of some tags that could not be dealt with during the revision stage. These carry out a series of checks, with error messages being sent to the linguist responsible for the document so that the appropriate corrections can be made.

Inclusion

By the time the documents reach this stage they comply with all the requirements for becoming part of the corpus, so they just have to be assigned to the directories that have previously been defined for this purpose.

Hardware, financial and human resources

In our specific case, this area is conditioned by the way in which the Centro Ramón Piñeiro para a Investigación en Humanidades is organised and run. The project normally works with eight people: the project director, the linguistic coordinator, the computer

[8]Amongst other reasons because not all the phenomena contemplated in the DTDs were marked during the structuring stage.

[9]E.g. because the OCR program makes mistakes in character recognition.

coordinator, five grant holders on the linguistic side and one grant holder on the computer side. The grant holders stay with the project for a maximum period of three years, and a selection process is instituted when they leave to replace them with new researchers.

This rotation means that it is extremely important to ensure that the initial training received by these grant holders is done in the shortest time possible, since the sooner they interiorise the work process and protocols the less delay there will be in the project.

Each member of the team has their own computer. At the time of writing the operating system installed in the computers used by the linguists is Windows XP[10], although we are currently considering migrating all the computers to a GNU/Linux environment [36, 53, 54].

The centre also has an additional computer connected to a scanner with automatic page feeder for tasks to be performed on hard copy documents during the acquisition stage, and a further two computers with a stable Debian operating system [81] which are used by the computer team to develop and execute the various transformation scripts.

Storage

For storage purposes there is a server that shares a hard disk unit with all the other computers used in the project. All the computers and members of the project connect to it as unit J:. This unit contains the following directories:

- corga_xml_acquisition: this contains the first digital version of the documents available (as obtained at the end of the acquisition stage)
- corga_xml_structuration: this includes the documents as they stand after the structuring stage
- corga_xml_conversion: this contains the documents after they have been processed by the scripts during the conversion stage.
- corga_xml_revision: this includes the documents currently at the revision stage.
- corga_xml_inclusion: this contains the definitive version of the documents, i.e. those that been validated as being error-free and now form part of the corpus.

A backup copy of the content of the hard disk on the server is made every day, using a backup robot capable of handling all this volume of information.

4.2 Checking the methodology

Once the methodology has been built, but before the whole team begins to work on the development of the corpus, it has to be tested. To this end, the linguistic and computer coordinators carried out the whole conversion process from beginning to end, i.e. from the acquisition stage to the inclusion stage, on one text for each type of document that had been defined.

This allowed us to detect and remedy the occasional error in the various conversion scripts and certain deficiencies and complications in the processing of the texts. Similarly, we were able to take note of the difficulties that arose during some of the revision operations and include these in the relevant protocols.

Once the methodology had been successfully checked for a given document type, we then moved on to the scheduling of the development of the corpus for the type of document concerned, a task that involved the full project team.

[10]Windows XP is a registered trademark of Microsoft Corporation.

4.3 Scheduling

The guiding rule followed by the project team when scheduling their work was, and still is, to attempt to maintain a balance between the following three parameters: lustra, thematic areas and media.

At present the scheduling is subject to annual revision, with a view to maintaining this balance between these three parameters as far as is possible. A further question that has to be particularly taken into account is the need to plan the acquisition of documents sufficiently in advance so as to avoid potential problems of unavailability of texts for processing.

4.4 Development

The development stage involves the full project team. New members to the team need to undergo a certain amount of training before they can become involved in the development stage. It requires an in-depth study of the processing protocols[11] and the need for the linguistic coordinator to review the outcome of the work done by a new member on processing documents during an initial period of time.

Once it has been verified that the work done by a new member of the team meets the required quality criteria, it is no longer revised. In any case, the validation scripts prevent a large number of errors from entering the corpus, which guarantees that texts comply with a reasonable level of quality. Furthermore, several sessions are held each year for the purpose of discovering hitherto undetected errors in documents.

Moreover, throughout the whole of the development stage the linguistics team hold weekly meetings to comment on the problems encountered during the revision of documents, the difficulties posed by specific texts and the complex casuistic relating to structuring or sentence segmentation. Individual cases are resolved on a consensus basis and the decisions are recorded in the processing protocols to avoid the need for the same discussions to be repeated in the future.

5 IR system requirements

In this section we look at the linguistic IR systems used with large-scale corpora in which documents are structured according to the XML standard and no morpho-syntactic information is included.

Of particular relevance, therefore, will be the frequency of occurrence of a given word and the display of specific examples in context, but it will also be necessary to be able to ensure that the performance of the system meets the expectations of its users, who will normally be linguists. Some of these corpora are extremely large, containing hundreds or thousands of documents that in turn contain tens or hundreds of million words, so the IR systems designed to work with them have to be sufficiently efficient and flexible.

Below we describe what in our view are the characteristics that should be found in most IR systems of this type, these being used in subsequent sections to determine to what extent the most representative of current technologies are capable of carrying them out:

1. **Full-text searches**: within the field of full-text searches, we can distinguish between:

[11] Hence the importance of having protocols, so that all the members of the team will be doing things in the same way.

(a) Exact match searches. In other words, they show the cases that coincide exactly with the search string.

(b) Accent sensitive or non-accent sensitive searches. These allow us to decide whether or not accents should be taken into account during the search.

(c) Upper case sensitive or non-upper case sensitive searches. These are similar to the above, but referring to upper and lower case characters.

(d) Character unification. This enables different characters to be unified, i.e. it does the same as an accent sensitive or non-accent sensitive search, but without a character having to be the same as another one with an accent.

(e) Boolean searches. These allow the use of Boolean operators, e.g. a search for the documents containing two specific words.

(f) Proximity searches. These search for words that are less than a given distance apart.

(g) Marked text exclusion searches. These enable users to ignore text fragments when performing a search. Words, expressions or even text fragments of greater length written in a language other than that of the corpus itself are tipically excluded using this option.

(h) Stop words. These enable certain frequently used words to be excluded from a search. Unlike other search systems, they are useful in linguistic searches to specify an empty list of stop words since information is frequently required relating to words often included in these lists.

(i) Multiple simultaneous search modes. Some technologies allow multiple simultaneous search modes from which the user can choose. Others, on the other hand, may offer the possibility of choosing from several search modes, but not simultaneously, and these have to be established a priori by the computer team.

(j) Configuration of the characters that form part of words. At times we may not want a character to form part of the words in the search index, but at others the opposite may be true. A case in point is the hyphen (-), which in the case of Galician language we want to form part of the word so as to search for verb forms with second form of the article, whilst in the case of Spanish we may on occasion do not want it in order to enable us to individually locate the two components of a compound word which use a hyphen to join them.

(k) Use of wildcards: this feature enables the user to insert wildcards or regular expressions in a query.

(l) Highlighting: this allows the user to highlight the word(s) that match a given query. As a rule, when using this kind of system to make a query, the user wishes to see the word(s) that match the query expression highlighted against those coming before and after, thereby making visual exploration of the results swifter and easier.

2. **Statistical capabilities**: this consists of obtaining numerical values on different levels, e.g. counting the number of both occurrences and documents that correspond to a given query.

3. **Additional information**: the system offers information about the results obtained for a given query, e.g. in addition to the occurrences that match the search criteria,

it also displays the name of the author or publisher, i.e. additional data obtained from the structure of the documents in the corpus.

4. **Context**: as well as the occurrences that match the search criteria, it should also be possible to display the context in which they occur. This feature, which is also known as Key Word in Context (KWIC), allows the user to see the *n* words to the left and right of the key word specified in the query.

5. **Section independence**: with this feature the user can restrict a search to specific sections of documents or even to areas marked within the text itself.

6. **Character sets**: systems should be able to operate appropriately with the character set(s) needed for the documents contained in the corpus, of which the most commonly used are the ISO [50] (ISO-8859) and Unicode [91] (UTF-8) standards.

7. **Result browsing**: this enables the user to browse amongst the various pages of results. IR tools often require some type of page browser and a way of preventing all the results from being returned at once. For this reason it is a positive advantage to have a mechanism that will obtain the required information page by page.

8. **Result ordering**: a feature that allows the user to simultaneously order search results according to one or more criteria. Another interesting option is that of ordering results alphabetically according to the strings that match the query when wildcards are inserted.

9. **Structural relationships**: a way of increasing the flexibility of systems so that structural relationships of a certain degree of complexity can be included in queries.

10. **Query refining**: this allows users to refine their queries, i.e. to make a new search on the results of a previous one.

11. **Result organising**: a feature that allows the user to configure how results are displayed, e.g. the number of occurrences per page.

12. **Result saving**: enables results to be saved for subsequent processing at local level.

13. **Efficiency**: the system should be efficient enough to leave users satisfied with its performance.

14. **Name list**: this allows the user to obtain data about the subset of documents that is being consulted.

6 Evaluating technologies

Although in our case we take as our starting point documents that have been structured with an XML standard, this does not necessarily mean that the technology used to develop the IR system has to be able to interact directly with this standard. An XML document can be converted to a wide range of different structures [22, 88, 102], and therefore a similarly wide range of technologies should be submitted to the evaluation procedure.

During this evaluation we covered the main technologies currently available on the market, with priority being given to free software alternatives. As our commercial point of reference we used Oracle [60], which as will become clear below, to a large extent covers all our requirements.

6.1 Text indexers

There are currently several text indexers on the market: Lucene [2], dtSearch [29], Swish-e [49, 69], Sphinx [82], etc. However, most of them focus on obtaining an ordered list of relevant documents relating to the query made, i.e. their main result is a set of documents on a given topic, organised according to their degree of relevance to the query. This means that they often make it rather difficult to cover certain of the aspects included in our proposed list of requirements, which is based on occurrences of key words rather than on whole documents.

6.2 Linguistic applications

A wide range of different applications were found in this category, amongst them: Tact [90], OCP [62], MonoConc [7], WordSmith tools [63] and Sketch Engine [51, 52], most of them monolithic pre-Internet applications that were therefore conceived without considering the currently popular network-based concept.

6.3 Relational databases

The development of an IR system using a relational database management system can be approached from a variety of angles. We have therefore classified all these possibilities in two alternatives: those that do not need to individually separate the words from the documents into a table and those that need to do so.

The basic structure for the first of these alternatives would be that consisting of a single table that includes both the data from the headings of the documents and the text of the documents themselves, whilst the second alternative would need at least one additional table for the words from the texts, related to the document table. Apart from these two basic structures there are many other intermediate ones that can be configured for each individual case, meaning that the questions presented here can be combined to provide a solution for any given case.

Section 2.3 of Part III of the doctoral thesis contains a detailed explanation of the possibilities of Oracle [60], MySQL [83] and PostgreSQL [68] for satisfying the requirements listed above.

6.4 XML management systems

We have left XML management systems until last, because although they might seem to be the most appropriate option for developing systems of this kind in which documents are already in XML format, in practice the opposite is true, since their technology is still very new and some of the functionalities that we demand from these systems have not yet been considered.

Perhaps the most representative example of this technology is Tamino [80]. A prior analysis to see if it was appropriate for the systems represented in this work was made [11], but it confirmed that some of the required aspects are not dealt with appropriately.

Other alternatives are also available as free software, such as eXist [57] or Xindice [5], none of which at present meet our proposed specifications. The underlying problem they all share is that they are very recent management systems that aim to provide solutions for much more basic IR systems, and thus their efficiency in relation to the aspects that concern us here is inferior to that of other systems we have already mentioned. Just to give one example, the first version of the XQuery standard [101], which seems to be one of the most appropriate alternatives for querying native XML databases, was published in

Characteristics	Oracle 11g	PostgreSQL 8.3	MySQL 5.1
1a	Yes	1st(1 term), 2nd(No)	Yes
1b	Yes	Yes	Yes
1c	Yes	+	Yes
1d	+, <= 2 characters	Yes	+
1e	Yes	Yes	Yes
1f	Yes	1st(No), 2nd(Yes)	1st(No), 2nd(Yes)
1g	Yes	+	+
1h	Yes	Yes	Yes(global)
1i	Yes	Yes	duplication
1j	Yes/Yes	Yes/+	Yes(global)/+
1k	%,_	%(not suffixes)	*(not suffixes)
1l	1st(Yes: -size), 2nd(Yes)	Yes	1st(No), 2nd(Yes)
2	1st(Yes: -words), 2nd(Yes)	1st(Yes -words), 2nd(Yes)	1st(Yes -words), 2nd(Yes)
3	Yes	Yes	Yes
4	1st(Yes: -size), 2nd(Yes)	Yes	1st(No), 2nd(Yes)
5	Yes	Only structured	Only structured
6	Yes	Yes	Yes
7	Application cache	Application cache	Application cache
8	1st(Yes:-words), 2nd(Yes)	1st(Yes -words), 2nd(Yes)	1st(Yes -words), 2nd(Yes)
9	Yes	Yes	Yes
10	Yes(repeat)	Yes(repeat)	Yes(repeat)
11	Yes	Yes	Yes
14	Yes	Yes	Yes

Figure 3: Comparative fulfilment of proposed requirements

2007, so it will still be some time before it will be able to provide efficient support for the kind of queries we have in mind.

6.5 Comparison

This section provides a joint analysis of the possibilities offered by all the selected technologies in order to provide an overview and determine their suitability for the IR systems we are talking about.

Firstly, the table in figure 3[12] gives a joint overview of the possibilities of the various relational database management systems under consideration. We can see at first glance that the Oracle column is the one with the greatest number of unconditional yeses and not even a single no. Next we provide a comparative analysis under a series of relevant headings:

Full-text searches

As can be seen from the table, the only limitations of Oracle with regard to full-text searches are that it only allows for the unification of up to two characters, which is usually sufficient in most cases, and it does not allow the user to define the size of the context when the text is not structured into words.

On the other hand, with PostgreSQL the user is unable to perform multi-word searches, a very basic requirement of the systems under consideration. Neither does it allow for proximity searches unless the text is divided into words, nor queries for word prefixes/suffixes, and it is no easy matter to exclude characters from the words in the

[12]The use of the + symbol indicates that more work than expected will be needed to develop this characteristic. This includes any work needed to program more than the stablishment of simple configuration parameters. Similarly, the references 1st or 2nd refer to the first and second alternatives, respectively, mentioned in the previous section, namely whether words are not structured independently or otherwise.

index. Furthermore, a new parser has to be developed in C language in order to perform upper case-sensitive searches, and excluding tagged text from the searches requires a great deal of extra work.

Finally, MySQL demands a lot of extra work for character unification to be carried out, for excluding tagged text from queries and for ignoring characters from words when building indices. Furthermore, the list of stop words and the configuration of the characters that can form part of the words are configured globally for the management system as a whole, which makes it impossible to have different configurations for a variety of IR systems. Nor does it allow various text indices to be built for a single column of data without duplicating their information, and in the case of the first alternative it does not allow proximity searches or the highlighting of text that matches a given query.

Statistical capabilities

None of the three technologies orders results by words that match the search term unless they are structured separately, and all three allow the user to do so when the words are separate.

Context

Only PostgreSQL allows the user to define the context size when the text is unstructured, but all three allow this to be done in the second case.

Section independence

Only Oracle allows searches to be made within marked sections in unstructured text with little effort.

Result browsing

Due to the potential size of search results, in all three technologies it is recommendable to have a cache in the search application that enables us to browse the results page by page without having to repeat the query.

Result ordering

None of the three technologies allows results to be ordered by strings matching the search in the first assumption, and therefore in this case the words will have to be entered independently in a table.

Query refining

In all three cases it is recommended to repeat the query with more restrictive filter criteria rather than work on a subset of results. This is because the initial result might be quite large and it may be less efficient to manage it than to repeat the query.

In conclusion we can say that, as shown in Figure 3, Oracle is the relational database management system that initially presents the fewest problems and, in the case of difficulties, offers the greatest number of facilities for resolving them in a reasonable period of time. As a result, despite the considerable cost of acquiring an Oracle licence for production environment, it is currently the technology of choice if problems are to be avoided in the development of environments of this nature. Furthermore, Oracle is very

Tecnology	**Main reason**
WordSmith tools	Monolithic application
Sketch Engine	Lack of high level filters
DtSearch Swish-e	Poor document structure
Lucene	Designed to obtain documents
Sphinx	Experimental
Tamino eXist	Poor performance
Xindice	Experimental

Figure 4: Rejected technologies

close to remove the need to use a word table for this type of system. However, it would appear that it will not be long before free software alternatives will be able to rival the performance of proprietary tools.

Figure 4, on the other hand, shows the main reasons why the other technologies under consideration were considered unsuitable for dealing with the problems we propose. On the whole, none of these technologies currently offers an acceptable level of search possibilities that correspond to our requirements, although in this case some of them are making very rapid progress. So, it is important to keep track of them in the view of possible evolutions that will make them more competitive.

6.6 Performance

We now evaluate the performance of the three selected relational database management technologies (Oracle, PostgreSQL and MySQL), using as our bench the basic database structure developed for the CORGA project, shown in Figure 5. Although the design principles will be analysed in the following section, we will now explain the basic concepts in order to make the evaluation clear.

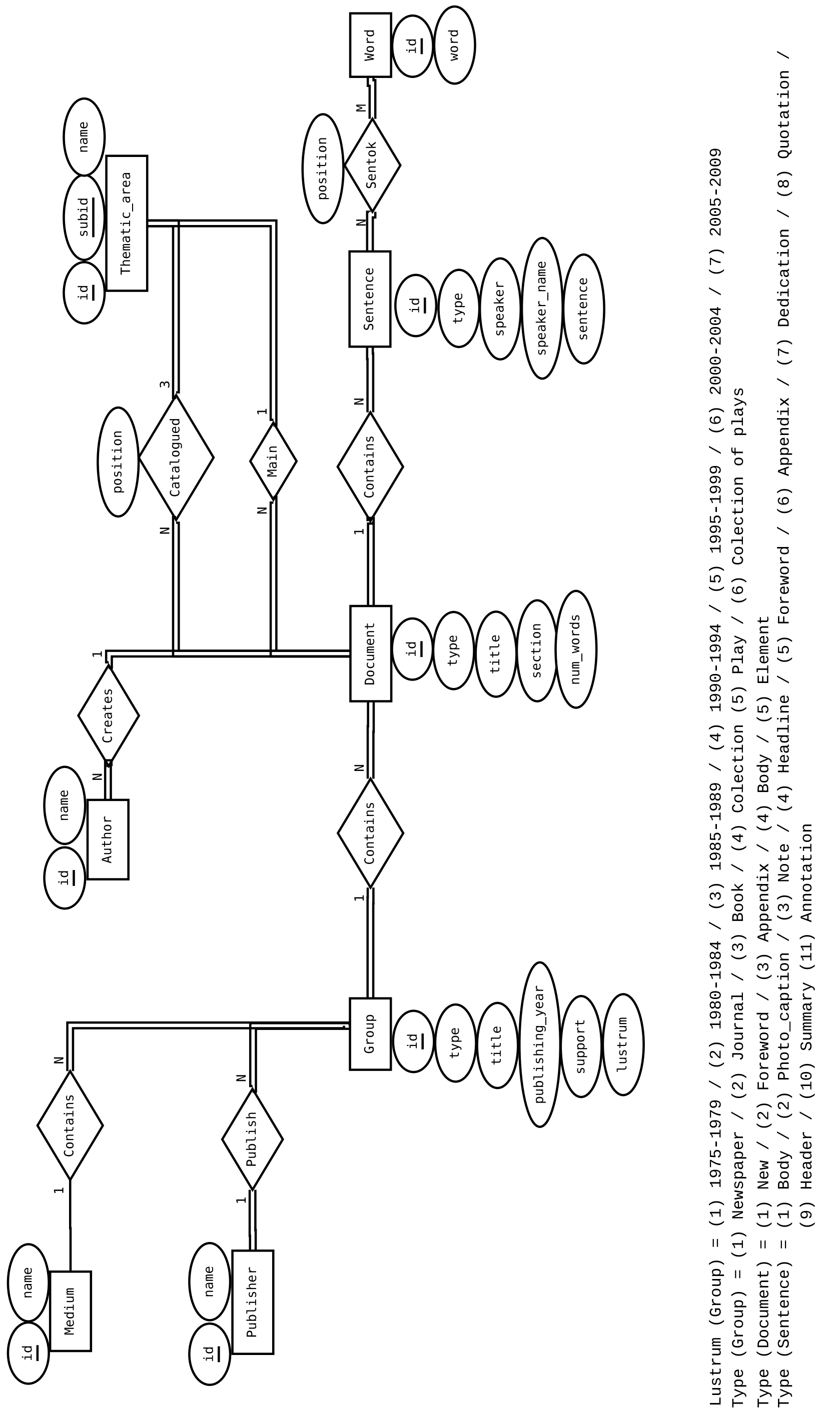
As can be seen from the figure, the core of an XML document is represented by two entities: Group and Document. For example, in the case of a newspaper, the entity Group represents the whole newspaper, whilst the entity Document represents the news it contains. Each document is assigned its sentences, and finally each sentence is associated with its component words by means of an N:M relationship that stores the position of the word within the sentence. Additionally, the documents have a medium of publication, a publisher and authors, and are catalogued in up to a maximum of three thematic areas. The number of words (num_words) each document contains is also stored in order to meet the requirements of name list queries.

For the purposes of our evaluation we will therefore define several model queries that cover the aspects analysed above, which will be executed on each of the technologies hosting version 1.3 of the CORGA project, which includes 23,059,543 words distributed in 1,366,257 sentences, these in turn being distributed in 437 newspapers, 86 journals or journals and 390 books.

6.6.1 Model queries

We now define the queries that will be used during the evaluation process:

Q1a: Obtain the number of documents and sentences that match the expression *desenrolo* (=*development*).



Lustrum (Group) = (1) 1975-1979 / (2) 1980-1984 / (3) 1985-1989 / (4) 1990-1994 / (5) 1995-1999 / (6) 2000-2004 / (7) 2005-2009
Type (Group) = (1) Newspaper / (2) Journal / (3) Book / (4) Colection (5) Play / (6) Colection of plays
Type (Document) = (1) New / (2) Foreword / (3) Appendix / (4) Body / (5) Element
Type (Sentence) = (1) Body / (2) Photo_caption / (3) Note / (4) Headline / (5) Foreword / (6) Appendix / (7) Dedication / (8) Quotation / (9) Header / (10) Summary (11) Annotation

Figure 5: Entity-relation model of the CORGA database structure

Q1b: The same as the above, but searching for the prefix *des-* (=*dis-*). The purpose is to see the cost of a search for a prefix.

Q1c: The same, but for the suffix *-ado* (=*-ed*).

Q2a: Obtain the number of documents and occurrences that match the expression *desenrolo*.

Q2b: Idem, for the prefix *des-*.

Q2c: Idem, for the suffix -*ado*.

Q3a: Obtain the number of documents and sentences that match the expression *sen embargo* (=*however*).

Q3b: Obtain the number of documents and occurrences that match the expression *sen embargo*.

Q4aa: Obtain the number of documents and sentences that match the expression *sen embargo* within the thematic area *Economía e política* (= *Economy and politics*).

Q4ab: Obtain the number of documents and sentences that match the expression *sen embargo* in the articles in journals.

Q4ac: Obtain the number of documents and sentences that match the expression *sen embargo* from documents whose author is a person with *Rivas* in their name.

Q4ba: As in Q4aa, but obtaining the number of occurrences instead of sentences.

Q4bb: As in Q4ab, but obtaining the number of occurrences instead of sentences.

Q4bc: As in Q4ac, but obtaining the number of occurrences instead of sentences.

Q4ca: Obtain the number of documents and sentences that match the expression *sen embargo* in news items from journals with the thematic area *Economía e Política* and whose author is a person with *Rivas* in their name.

Q4cb: As in the previous query, but obtaining the number of occurrences instead of sentences.

Q5a: Obtain the number of documents and sentences that contain the expression *sen embargo* for each type of medium, main thematic area and lustrum. Since there are 3 types of medium, 6 main thematic areas and 7 lustra, the total number of different values to be calculated is 32.

Q5b: As in the previous query, but obtaining the number of occurrences instead of the number of sentences.

Q6a: Obtain the sentences in which the expression *sen embargo* appears.

Q6b: Repeat the previous query, but also obtaining the title of the documents in which it appears, the publisher, the authors, the medium, the thematic areas, the year of publication and the type of sentence.

Q7a: Repeat query Q6a with the word *desenvolvemento* (=*development*).

Q7b: Repeat query Q6b with the word *desenvolvemento*, ordering the result by year of publication, medium and main thematic area.

Q8a: Obtain the number of words and documents in the complete database

Q8b: Obtain the number of words and documents relating to journals with the thematic area *Economía e Política*.

Q8c: Obtain the number of words and documents for each medium, main thematic area and lustrum (32 different values).

Q8d: As in the previous query, but only for documents from the thematic area *Economía e Política*.

6.6.2 Hardware and operating system

The hardware used during the tests was a midrange Hewlett-Packard ProlLiant server [48] with the following features:

Model: HP Prolliant ML310 G4

CPU: Dual-Core Intel Xeon 2,13 GHz

Bus: 1066 MHz

Cache: 2 Mb L2

RAM: 3Gb

Hard disk: 136 Gb SAS at 15.000 rpm

Operating system: CentOS 5.2

The operating system chosen was the 64-bit version of CentOS 5.2 [19], basically because of the ease with which Oracle can be installed in this system and because it has been designed as a serious alternative for servers. It would therefore also be a feasible option if we wanted to install the system in a production server rather than one specifically dedicated to testing, as in our case. A minimum graphics and services configuration was set up on the server to reduce as far as possible the impact of other processes on query performance.

6.6.3 Evaluation

Users of systems of this kind rarely make the same query in a short period of time. So, in order to obtain the results for the various queries from the batteries defined above each query was performed on three separate occasions, just after the management system had been started up, to avoid the problem of response times being influenced by the cache memory of the technology being evaluated. The final result was calculated as the mean of theses three values.

As can be seen in Figure 6, which shows mean response times in seconds for each query, Oracle is the management system that returns the best response times, with a high number of queries having a response time of less than one second.

The response times in bold are those that are clearly unacceptable for the kind of system we are interested in, whilst the symbol x means that no response could be obtained from the system for the query in question[13]. We can see how even Oracle had problems responding to some of the queries proposed. A more detailed analysis of the results leads to the following conclusions:

1. Access to prefixes and suffixes, which return a large number of entries in the text indices, creates problems for all the management systems (queries Q1b and Q1c). In our opinion this is because the integration of text indices in relational database systems is still not particularly well developed.

[13]This may have been for one of two reasons: either the response took more than 5 minutes or the management system is not capable of performing this kind of query.

2. We found that in general it took longer to calculate the number of occurrences as opposed to the number of sentences, this being mainly due to the `JOIN` operation that has to be performed with the words table. Proof of this are the results of queries Q3b and Q5b compared to those for Q3a and Q5a, respectively.

3. Since PostgreSQL does not allow multi-word queries on the text index, a functionality that is being developed for the version 8.4, the word table not only needs to be used when consulting occurrences, but also when obtaining values for sentences (Q3a, Q4aa, etc.). This means that even though acceptable values may be obtained for expressions consisting of one or two words the response times get worse as the expressions get longer.

   Furthermore, the fact that PostgreSQL does not allow the use of wildcards in its text search expressions is a major handicap for a system of this kind.

4. The results would also appear to indicate that the query planner and optimiser of the various management systems does not perform as well as it should with regard to accessing text indices, which means that on certain occasions response times are disproportionately high.

   This is the case of MySQL for queries Q4bc, Q4ca and Q4cb, where it seems that when accessing the text indices corresponding to the table of authors, to filter by author name, and that of the query expression on sentences, the planner is unable to properly prioritise and/or optimise access to these test indices. This type of problem was also detected in PostgreSQL (Q5a and Q6b) and in earlier versions of Oracle than the one evaluated here.

The conclusion that can be reached from these performance tests is that although on the whole Oracle shows the best response times, this technology also encounters problems when trying to answer certain types of query that are vitally important in environments of this nature.

The open licence management systems appear not to be too far away from equalling the performance of proprietary systems, but it is fair to say that neither the former nor the latter take into account the IR systems we are concerned with in this study, in which the cost of inputting or modifying data is of little relevance, the use of text indices is vital and redundancy may be one way of enhancing their performance.

7 The CORGA IR system

Before this study was undertaken the CORGA project was already equipped with an IR system that allowed users to make some of the searches presented in Section 5. This system used the dtSearch tool to search the document database in the previous text format used by the project, but was unable to carry out some of the kinds of search considered relevant to the project:

1. To incorporate the concept of sentence when displaying the context of an occurrence, rather than just displaying a certain number of words before and/or after the occurrence that match the query term.

2. To enable sequences of words in another language to be shown, even when queries do not take place on these elements.

Query	**Oracle 11g**	**PostgreSQL 8.3**	**MySQL 5.1**
Q1a	1	<1	<1
Q1b	**90**	x	**85**
Q1c	**50**	x	x
Q2a	1.5	4.5	2
Q2b	20	4.5	**191**
Q2c	x	x	x
Q3a	1	4.7	2
Q3b	11	x	**30**
Q4aa	<1	4.3	2
Q4ab	<1	4.3	<1
Q4ac	<1	**31**	7
Q4ba	<1	4	4
Q4bb	<1	4	2
Q4bc	<1	3.6	**64**
Q4ca	<1	5	**78**
Q4cb	<1	5	**64**
Q5a	1	**37.5**	9
Q5b	18	21	13
Q6a	<1	7.5	<1
Q6b	3	**60**	1
Q7a	<1	<1	<1
Q7b	<1	<1	<1
Q8a	<1	<1	<1
Q8b	<1	<1	<1
Q8c	<1	<1	<1
Q8d	<1	<1	<1

Figure 6: Mean response times in seconds for each test query and technology

3. To provide a search criterion that enables the user to determine specific sections of texts for a search, e.g. only newspaper headlines, news summaries, prologues, appendices, etc.
4. To include a degree of flexibility in the combination of different result ordering criteria, enabling several to be simultaneously combined.

For this reason it was decided to develop a new IR system using Oracle, in parallel with the document adaptation process, in which the knowledge acquired during the research for the thesis was applied.

7.1 Requirements analysis

The full set of requirements that the new system has to fulfil is as follows:

Basic query modes

- Exact phrase: to find the occurrences that match a given expression.
- Any word: to find any of the words in the expression.
- All words: to find the sentences that include all the words in the expression.
- Boolean: to find sentences that verify the Boolean expression defined in the query.

Additionally, the query modes need to allow the user to decide whether or not the search should be accent-sensitive or not.

Wildcards

It should also be possible to use two wildcards:

* to indicate the substitution of zero or more letters.

? to indicate the substitution of a single letter.

Search criteria

The search criteria include the following: thematic area and sub-area; time interval, from 1975 to the present day; medium, which could be newspaper, journal or book; and finally, document section, in which the desired search areas within the document type can be specified.

Statistics

The number of occurrences/sentences and documents can be calculated for the four basic query modes and the various search criteria specified.

The system also obtains statistics concerning media, lustra and main thematic areas. Thus, for media statistics, the values would include the number of sentences/occurrences and documents for newspapers, journals and/or books; in the case of lustra, data on sentences/occurrences and documents for the time intervals 1975-1979, 1980-1984, 1985-1989, 1990-1994, 1995-1999, 2000-2004 and 2005-2009; and in the case of thematic areas, data on sentences/occurrences and documents belonging to the thematic areas of economics and politics, culture and the arts, the social sciences, science and technology, fiction and others.

Context

The system displays the words that match the query expression, highlighting them within their context, the latter being given at sentence level. Thus, the initial list of results contains the full sentences in which the expression appears highlighted, whilst the system enables the user to subsequently expand the context to include the sentences coming before and/or after the initial one.

Result ordering

Results can be ordered by date, thematic area and medium. When ordered by thematic area and/or medium, the date is used as a complementary ordering criterion.

Text exclusions

The sequences marked in a text as belonging to another language are displayed differently when they appear in the context of the target query term. This means that it is impossible to obtain occurrences of words that match with the expression within these sections, that is, the words thus marked are excluded from queries.

Additional information

When an occurrence matching the query term is displayed, information is also given about the title of the document to which it belongs, the author, the publisher, the thematic area or areas to which the document belongs, the medium and the year of publication.

Performance

The system response times are satisfactory as far as the user is concerned, in most cases being no more than a few seconds.

Web application

The system can be accessed from a browser, i.e. it is a web application.

Name list

The application includes a system for searching the name list of publications which allow us to use the previously described search criteria. In other words, it is possible to obtain the total number of words and documents relating to the query criteria chosen, as well as the metadata related to the works selected.

7.2 Analysis and design

In this section we will describe the most relevant elements related with the analysis and design of the tool in question. It is not our aim here to deal with all the elements related to the analysis and design of an application, but rather to focus on those that specifically help us to understand the particular features of the one we are describing.

7.2.1 Application architecture

The application was developed in Java [41] programming language following the Model-View-Controller (MVC) pattern [39, 84]. Although there are currently various different working framework that have been developed around this pattern [3], the birth of our system took place before these projects became stable, and we therefore followed our own deploying scheme.

Analysis class diagram

Figure 7 shows the high level UML class diagram [16, 17] for the query system, which makes clear the distinction between the three different classes: controller, model and view.

We now describe the responsibilities of each of the analysis objects:

- **Controller, Agent and** JSP: the controller is responsible for receiving the requests from the browser, and delegates the appropriate actions to the agent. Once the agent has performed its function, ther former delegates the task of displaying the results of the query to the corresponding JSP.

  In order to make this communication between the controller and the JSP possible, the agent introduces into the HTTP session the objects needed to display the result and the values needed for future interventions by the user.

- **Session**: this class represents a user work session and materialises the query cache in such a way that when the user browses through the results of a query, the latter does not have to be repeated. This is why the Session object draws together the different objects needed to maintain the session information, i.e. the data needed to resolve successive requests made by the user.

- **Manager**: this class is responsible for creating the connections with the database and carrying out the queries on it.

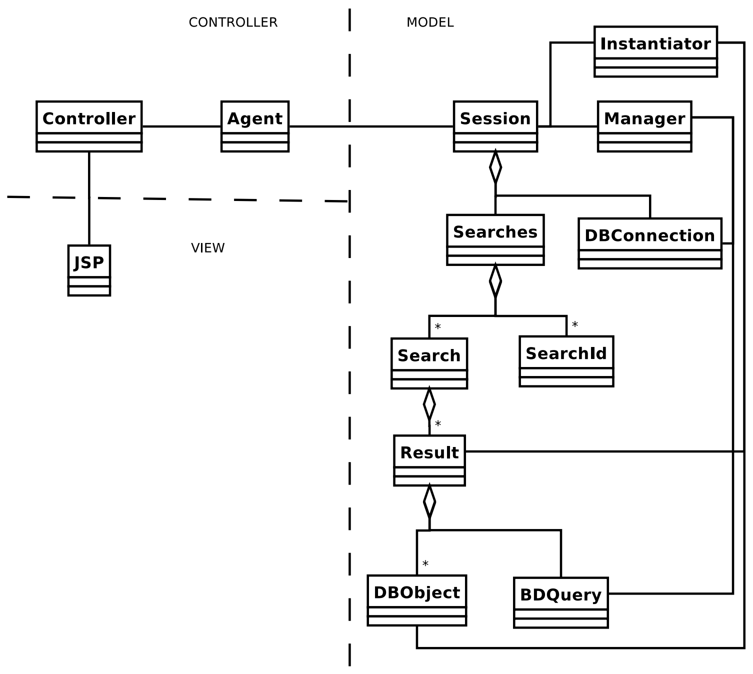


Figure 7: Analysis class diagram

- **DbConnection**: a class that represents the connection with the database. As a rule, each user has a connection associated with his work session.
- **Instantiator**: the class responsible for instantiating the objects resulting from a query.
- **Search and Result**: the Search object represents a user query and draws together the various results the query may deliver. In other words, there is a single search associated with a unique set of criteria defined by the user in the form, but this search may return more than one result.

  To give an example, a user initiates a search for a word and obtains the number of occurrences and documents containing that word. He then wants to see the occurrences in their context. This gives us a single query with two results, the initial one being a numerical value and the second the contexts in which the word appears.
- **Searches**: this brings together the various simultaneous searches that a user may be running. This occurs when a given user has various browsers open in which he is running different queries. There will be one Search object for each browser connected to the system. In our case we introduced a ceiling of five simultaneous queries per user.
- **DBObject and DBQuery**: a result may be constituted by one or more DBObject objects or by one DBQuery object.

  The first case occurs when the result of a search consists of a small set of values (units or tens), so the Result object stores the DBObjects that correspond to the individual results. If the user needs to obtain these data again for later queries, the application would not repeat the query because it already has them.

In the second case, however, the Result object retains a representation of the execution of the query (DBQuery). This occurs when the search result is a potentially large number of data (hundreds or thousands) that will need to be browsed through page by page. By keeping a representation of the query we will be able to access the various results pages without having to repeat it.

The class diagram for the name list subsystem is the same as that for the search system shown in Figure 7, with the only difference that the suffix *Namelist* needs to be added to the name of all the classes except that of the controller, which is shared by both subsystems.

Sequence diagrams

Section 4.2.1 of the thesis contains the sequence diagrams corresponding to the use cases *authenticate* and *formulate query*, which help to reveal the most relevant aspects of the application´s architecture.

7.2.2 Database

When dealing with large-scale structured corpora, the choice of structure for the database that will support all the queries from the system is one of the most complex aspects to resolve. A mid- or high-range server coupled with a top-ranking database management system can easily have difficulties in responding with ease and rapidity to the queries in systems such as the present one.

Although there are myriad possibilities for building the database, these can in broad terms be classified into two categories:

1. Those that give priority to unstructured content, i.e. the text itself contained in the documents. These are used, for example, by Mark Davies in various works [26, 27, 28].

   Systems of this nature usually deal very efficiently with purely linguistic queries, but do not allow the use of higher level criteria to delimit their scope. A partial solution to this problem is provided by the creation of parallel databases that contain pre-calculated data for different elements of interest, but they nevertheless continue to cause problems, because the criteria for which these values can be obtained are defined a priori by the person developing the system rather than a posteriori by the user, which would be preferable.

2. Those that place text and document structure on the same level, this being the approach we defend in this study. Although the response to certain types of query may be slower than in the first case, they are much more flexible and users can define search criteria that include structural and text metainformation elements.

In the present study, we therefore focus on the possibility of handling metainformation about the texts, which will logically have a knock-on effect on the structure chosen for the database.

Reciprocating what was said about document structure in Section 3.1, neither should the database structure be exclusively conditioned by that of the documents. We obviously need to take the latter into account when building the data model, but we should not be overly influenced by the rich and flexible structure of the texts, since we would otherwise develop a highly inefficient system.

In the case of the CORGA project, which includes various types of document (newspapers, journals, books, collections, plays and collections of plays), it would be unfeasible to have a separate set of tables for each type of document. Even though the search system would be extremely flexible if this were the case, we would need to perform a separate query for each type of document when consulting the whole database, which even in the best of circumstances would be a highly inefficient procedure. There may even be problems in displaying a single set of results (irrespective of document type).

The most appropriate course of action in this case is therefore to extract common factors from the different text typologies and build a shared structure that agglutinates the relevant aspects of the various types of document. In all likelihood there will thus be the occasional highly specific feature that will not fit and an element of flexibility that will have to be sacrificed, but these will presumably be relatively minor in terms of number and importance, whilst the resulting system will be much more efficient and still extremely flexible.

The entity-relation model for the application is that used in the evaluation process and shown in Figure 5, which we describe in detail below.

Document representation in the database

The basic document representation structure consists of the following entities: Group, Document and Sentence, with document metadata being stored in the first two of these.

Thus, in the case of a newspaper or journal, the metadata about the publication would be in the entity Group, whilst those about the different news items would be in the entity Document. In the case of a book, the data would be in Group and would be repeated in Document. Additionally, a book may have forewords and appendices by other authors whose metadata would also be included in Document. Finally, in the case of collections the elements concerning the collection itself would be stored in Group, whilst the metadata associated with its different components would be located in Document.

This way of organising the information means that queries are made against a single database, thereby optimising response time for the most frequent queries, namely those that act on the complete set of documents.

Furthermore, since what constitutes a document in this entity-relation model may have up to three thematic areas, and that it may at times be necessary to order results according to this criterion, we will need an additional relation between Document and Thematic_area. This represents the association between documents and their main thematic area and prevents the repetition of documents in the results.

Mention should also be made of the attributes *num_words* in Document, which includes the number of words in the documents to support name list queries, and *section*, where the sections of the newspapers in which the news items appear are stored. In the latter case we have been able to include specific information about a single type of text whilst still using only one database for the complete text collection.

Text components

The entity Sentence accommodates the different sentences in the documents and includes, in addition to the text of the sentence itself, the document zone in which it appears[14] and, in the case of plays or conversations, the name of the character or speaker

[14]This is the field *type*, which contains numerical values that represent the following zones: body of the document, photograph caption, footnote, news headline, etc.

in question[15]. In the case of interviews reference is also made as to whether the sentence is spoken by the interviewer or the interviewee[16].

For its part, the entity Word contains the words in the sentences in the text. These are connected with the sentences in which they appear by the relation Sentok, which includes information about the position of each word in the sentence. This relation is used in queries that need to use the attribute position in multi-word expressions.

Materialisation in tables

An analysis of the entity-relation model presented here needs to focus on various aspects of how the said model is materialised in its underlying relational schema. As can be imagined, particular attention must be paid to include certain basic aspects such as generating the various database indices, especially those that correspond to the foraneous keys and other tables that enable the joins to be made efficiently. However, there are other and less trivial aspects that should also be taken into account:

(a) Weak entities

Firstly, we have seen that in our model the entities Document and Sentence have been created as strong entities. The natural approach would be to think that Document is a weak entity dependent on Group, and that Sentence is in turn dependent on Document. The structure would thus be materialised in three tables: Group, Document and Sentence. The second of these would include as one of its fields the identifier of Group, whilst the third would have two fields corresponding to the identifiers of Group and Document, respectively.

The problem with this approach is that of the possible appearance of tuples that are repeated in different queries, mainly as a result of the fact that a document may have up to three different thematic areas, and therefore the `SQL DISTINCT` clause plays an important role in returning the desired results as efficiently as possible. Since this clause can only receive a single column as its parameter, queries are much easier to make if these entities are considered to be strong, incorporating a single field with the primary key of each one.

(b) N:M relation with the table Word

Secondly, we have seen that the relation between the entities Sentence and Word, called Sentok, has N:M cardinality, and therefore the table Word will include a single tuple for each different word in the text. Doing things this way has the advantage that the text index of this table is much smaller than it would otherwise be, making queries on it much more efficient. On the other hand, however, the management system then has to perform the join with Sentok and Sentence to enable queries to be made.

If this relation were to be materialised by means of a 1:N cardinality, the table Word would include a word for each time the same word occurred in the text. This would make the table in question much larger, any search of the text index therefore being much slower, but the join with Sentence would be much simpler.

After several tests in which both alternatives were used no clear difference was found between either of them, since the cost of performing the join in the first case is similar to

[15] *speaker_name* field.
[16] *speaker* field.

that of consulting the text index in the second.

(c) Justification of the table Word2

With the solution shown in Figure 5, there is a problem when a search for occurrences of an expression consisting of exactly two words is made. In order to perform searches of this nature, the management system has to access the tables Word and Sentok twice so as to compare the attribute *position* of the first word in the expression with *position+1* of the second one. However, this may take an unacceptably long time, for three reasons:

1. At least one of the two components of the two word expression is very often a frequently occurring word, thus giving rise to a large number of tuples from the join between Sentok and Word that match the text search.

2. Even though the filters act on the text index of the table Sentence to filter the full two-word expression, it is also highly probable that this expression will verify a large number of tuples from the Sentence table.

3. Making the `JOIN` with Sentok, which is a table with a large number of tuples (one for each word in the corpus), twice has a high cost and the comparison of positions, even if functional indices are used, is not as efficient as might be expected.

Taken together, these three reasons mean, as we have said, that these queries may be excessively inefficient. To overcome this problem we have created an additional table, called Word2, which contains the different sequences of two words and is related with Sentence by means of a new association (Sentok2). This means that we only have to use Sentok2 once instead of having to use the attributes *position*.

The problem does not occur in expressions of three or more words, which occur much less frequently than those of lesser length. In this case, if the search uses the text index of the Sentence table to locate the complete expression there will be a considerable reduction in the expansion of the join with Sentok and Sentence and the search will hence be performed within a reasonable length of time.

The same is true of searches for a single term, in which there is no need to use the attribute *position* and Sentok and Word are only used once, meaning that searches can be performed efficiently.

Treatment of the attribute *position*

The materialisation of the attribute *position* of the table Sentok takes the form of non-repeated consecutive numbers, i.e. these are not numbers that begin with 1 in each sentence and end in the number corresponding to the total number of words in the sentence, but rather continue to increase from sentence to sentence.

The advantage of this approach is that when *position* is used to make comparisons, there will only be one tuple in the table that verifies the value of this attribute. A unique index (`UNIQUE`) can be defined on this column, thereby optimising access to the information.

Removing unnecessary joins

Finally, mention should be made of an aspect that is related both to the relational model and the search application: particular care needs to be taken not to use unnecessary tables and joins when performing queries on the database. As a rule the object responsible for making the latter (in our case Manager) implements a generic system that may tend to use an excessive number of tables to perform searches. We have to pay attention to avoid this.

As can be seen, in practice it can at times be useful to get away from the academic concepts of relational models, although one must always be aware of how and why we are modifying them. In other types of system these optimisations may be an option, but in our case they are essential, since the system may otherwise fail to perform the function for which it is designed.

7.3 Development and implementation

Although the hardware used in the tests in Section 6 was that described above, the system currently in operation [21] is running on a slightly older computer, which we describe below:

Model: AlphaServer ES40 68/833 M1

CPU: 2 at 833 MHz

Bus: 100 MHz

RAM: 3Gb

Hard disk: 8 x 36,4 Gb SCSI disks at 15.000 rpm.

Operating system: TRUE64 Unix 5.1B

Furthermore, the version of the Oracle database management system in this server is the 9ir2 and, since the 11g version is not available for this platform and the 9ir2 version has some problems with the functionality of the systems presented here, this hardware will in all probability be updated in the near future. Nevertheless, response times for most queries made through the system are acceptable, and the equipment has now been providing a reasonably satisfactory service for some years.

The web application was developed using Java 1.5 [86] as the programming language, JDBC [85] for accessing the database and Tomcat 5.5 [4] for providing the service.

8 Conclusions and future work

During the research presented in this study we have dealt with various topics relating to corpus linguistics, but more specifically with the creation and exploitation of large-scale text corpora.

With regard to corpus construction, many works only deal with the issue of a single corpus, and there is very little information available regarding the necessary stages that a document has to pass through from its appearance in its original medium until it is included in the corpus. What is more, if we are talking about studies in which the aim is to generalise the processes in each particular case so that they can be used in projects with common characteristics, there is practically no information available whatsoever.

Between the time when corpora first started to be built and the present day, computer systems have evolved considerably, whilst the appearance of studies on the generalisation of corpus building processes has been extremely sparse. The most representative example dates back to 1992 [9], and since then there have been practically no attempts to define a new methodology.

As for the development of IR systems we focus our efforts on the study of those that work with large-scale corpora, since in the case of small-scale databases almost anything is possible and the technological limitations referred to here do not appear.

There is a variety of different standpoints regarding the technologies and how data are structured in order to build IR systems that allow queries to be made on a corpus. Some studies approach the problem by giving priority to accessing the lowest level information (i.e. the text itself) to the detriment of the highest level information (metainformation, structural components, etc.), whilst in our case we attach the same degree of importance to both kinds of information.

Bearing in mind these two fronts, we therefore deal with the topic from a global viewpoint, that is, we propose a specific methodology for constructing corpora whilst at the same time determining the most appropriate technology for developing query systems, in all cases illustrated with examples from a specific corpus: the CORGA.

8.1 Conclusions

With regard to the proposal for a methodology that can be used to build structured corpora, we place particular emphasis on generalising the different processes that a document has to go through before it is incorporated into a corpus, precisely due to this lack of recent studies on this topic. The outcome is a proposed methodology that we hope will prove useful for other projects.

This proposal is also accompanied by its concrete application to the CORGA project, which enriched the study with a practical viewpoint that complements the more abstract concepts. Although it lies outside the scope of a discussion of this research, it is interesting to note that the fact that we have used this same methodology in other projects with satisfactory results, which has proved to us that it is in fact sufficiently versatile to be used in different contexts.

On the one hand we have a wide variety of alternative approaches to corpus construction, each with its benefits and drawbacks, and on the other, the same can be said of the technologies that can be used. A host of different alternatives is therefore currently available for dealing with the content presented in this study, but we believe that the path set out here is a promising one.

The methodology we propose is based on state-of-the-art standard technologies (which are also producing successful results in areas others than the one dealt with here), uses simple tools, promotes documentation and defines the scopes of interdepartmental communication. With these four components we have succeeded in ensuring that the cost of putting this methodology into practice in a new project will not be excessive, when compared to the benefits it brings: standardisation, error detection, consistency, documented protocols, etc. This also guarantees a better evolution of the corpus over time, with changes being more foreseeable and less traumatic, and, to put it briefly, that we will have a higher-quality resource.

The second part of our research, however, deals with the development of IR systems that use the corpus as their data source. In this part we also tried to generalise a requirements analysis, and to this end we used information from other query systems that we have developed as well as the CORGA itself.

The number of technologies that a priori could be considered as candidates for the development of this type of system is absolutely unimaginable, so we had to draw up an inventory and produce a subsequent classification. We finally decided to group them into text indexers, linguistic applications, relational databases and XML management systems, and then attempted to evaluate which of them offered the greatest possibilities within a

GNU/Linux environment to provide a solution based completely on free software, which would provide a significant reduction in costs.

In any case, it is true that the free sofware based technologies tested have yet to prove their worth against other proprietary alternatives, at least with regard to the aspects we require. Even so, what most surprised us was that even in the case of proprietary software we encountered more difficulties than initially expected.

From a purely technical point of view there do not appear to be as many problems as were evidenced by all the technologies tested, since aspects that are really quite well covered by some are the problem in others, and vice-versa.

In our opinion, this occurs for two reasons. On the one hand, the development of the functionality that provides the basis for queries in IR systems of this nature is a relatively recent phenomenon, meaning that there is still a large amount of room for improvement, whilst on the other the requirements that are essential in the tools studied here are not a priority goal for the technologies analysed. These needs are clearly less in demand than others (bank transactions, invoicing, management, etc.), and thus the development priorities in these projects never match those of systems for exploiting corpora.

Finally, we also hope that the information we provide on the analysis schema for the CORGA IR system will also prove useful. In this case, the use of design patterns and the adherence to current trends in object-oriented programming enable us to work with a structure that is clear and easy to maintain.

8.2 Future work

We hope to apply the proposed methodology to new projects. Since a methodology should never be static, future cases of its application will contribute to revealing aspects that have not been covered in this study. We will therefore have to continually update, review and improve our proposal to include elements that have either not been considered here, or do not fully fit into the schema we present. The ultimate goal of this process of dynamic adaptation is to incorporate new requirements as they arise into both the methodological materialisation of the concrete case and the abstract theory.

But in addition to evolving this methodology in terms of expanding its radius of action, making it valid for an increasing number of types of corpus, we will also need to adapt it to emerging technological trends. Given the rate at which new technologies appear to take the place of others that swiftly become obsolete, we will have to remain on the alert to these changes to see how they may affect the methodology, and update it when necessary.

Similarly, and with regard to the development of IR systems, it is essential to monitor not only the evolution of the tools that have already been tested but also that of others that may arise in the future, to see to what extent they may be able to provide solutions for the problems outlined here. We will pay particular attention to those based on free software, which as we have already said can lead to considerable cost savings in the development and use of this type of system.

In this regard, it would be productive for us to become directly involved in the development of the free software technology that we consider to be the most promising. It is our belief, for example, that the use of text indices based on finite automata or suffix arrays, combined with index redundancy to improve the performance of the various types of access to them, may help to achieve some of the intended objectives.

Finally, the extension of this study to the field of annotated corpora will be another of the main lines of work to be undertaken in the future. In fact we are already making some progress in this direction, by establishing additional document processing stages, selecting the tools that will allow us to carry these out, studying the particular requirements of systems of this kind and evaluating different technologies that will enable us to meet these new needs.

As can be seen, there are still many fronts open on which progress can be made in order to improve the search potential of corpus-based IR systems, and we therefore trust that this research project will be the first of many that will continue down this path that we have opened up.