UNIVERSITY OF GRANADA

Departament of Computer Science and
Artificial Intelligence

PhD Program:
Probabilistic Models for Artificial Intelligence
and Data Mining

PhD Thesis Dissertation:

DESIGN AND EVALUATION OF NEW XML RETRIEVAL METHODS AND THEIR APPLICATION TO PARLIAMENTARY DOCUMENTS

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of Granada as partial fulfilment of the requirements for the European PhD degree in Computer Science

PhD Student:
Carlos Jesús Martín Dancausa

Advisors:
Prof. Dr. Luis M. De Campos Ibáñez,
Prof. Dr. Juan F. Huete Guadix and
Prof. Dr. Juan M. Fernández-Luna

Editor: Editorial de la Universidad de Granada
Autor: Carlos Jesús Martín Dancausa
D.L.: GR 898-2012
ISBN: 978-84-694-6937-8

Agradecimientos

Quiero aprovechar estas páginas para agradecer a todas las personas que han hecho posible la realización de esta tesis, ya que no ha sido un camino de rosas y sin ellos, esto no habría sido posible.

En primer lugar, quiero mencionar a Miren, mi mujer, porque es mi compañera de viaje en todos los sentidos y en este proyecto, como siempre, ha estado en todo momento a mi lado. Gracias por estar conmigo noche y día, por hacer que esta hazaña se haya hecho realidad, ya que ha habido muchos momentos de agobio en los que creía que no podía continuar, pero siempre estabas ahí para alentarme. Eres increible.

Por otro lado, están mi madre y mi hermana que han sido siempre un apoyo incondicional y gracias a ellas, junto con mi padre, han hecho grandes esfuerzos para que yo haya llegado hasta aquí. Sois una familia maravillosa y nunca os podré agradecer lo suficiente todo lo que haceis por mi.

Acabo de mencionar a una de las personas de las que más me he acordado estos últimos cinco años, mi padre. He recordado muchas veces aquel momento cuando te dije que ya era ingeniero en informática, sin embargo, ahora no estás aquí para poder darte otra buena noticia que se que te hubiera encantado. De todas formas, se que me estás viendo desde ahí arriba y estarás disfrutando como el que más. Que sepas que he sentido tu apoyo en todo momento y ni te imaginas como te lo agradezco.

Para continuar, quiero mencionar a mis familiares, mis cuñados, tíos, primos y a mis dos sobrinos, Nacho y Jesús (que bien me lo paso con estos enanos), ya que gracias a Dios soy muy afortunado de disfrutar de una familia muy unida que ha estado siempre pendiente de mí y dándome su aliento. También, me gustaría recordar a los distintos familiares que no me pueden acompañar en la finalización de esta tesis pero que siempre han sido para mi una fuente de inspiración o me han apoyado en todo momento. Gracias Maria del Carmen, tío Andrés y tío Gabriel, os echo de menos.

No menos importantes son mis directores de tesis D. Luis Miguel de Campos, D. Juan Manuel Fernández y D. Juán Huete, ya que sin ellos esta tesis no sería

posible y han tenido la paciencia de aguantarme durante estos cinco años. Gracias por vuestras enseñanzas, paciencia, ayuda y disponibilidad siempre que os he necesitado.

No quería olvidar mencionar a los miembros del servicio de documentación del Parlamento de Andalucía, en especial a D. Carmen Tur y D. Antonio Tagua, debido a que han sido una parte fundamental en esta tesis. Gracias por todo el tiempo que habeis dedicado, de vuestra apretada agenda, en explicarme los diferentes aspectos de la colección del Parlamento y por abrirme las puertas siempre que os hemos necesitado. Da gusto trabajar con un equipo así.

Tambien, quería recordar a mis compañeros del despacho 14 y alrededores (Javi, Borja, Soto, Palao, Mariló, Carlos, Sergio, Pedro, Fernandos, Andrés, Antonio, Alberto, Jesús, Martita, Nacho, Cora,... siento si se me olvida alguno) ya que hemos pasado muy buenos momentos juntos e hicieron que mi estancia en Granada fuera espectacular en todas las facetas. En especial, quería agradecer a mis compañeros de grupo y amigos, Miguel y Alfonso, su ayuda y colaboración en muchas partes de esta tesis.

Por último, quería agradecer a los miembros del Informatics Institute de la Universidad de Amsterdam, en especial al profesor D. Maarten Marx su increible acogida durante los tres meses de estancia. Guardo muy buenos recuerdos de esta experiencia.

Esta tesis doctoral ha sido financiada por los proyectos de la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía P09-TIC-4526 y TIC-276, por el proyecto del programa de investigación Consolider Ingenio 2010: MIPRCV (CSD2007-00018) y por el proyecto del Ministerio de Ciencia e Innovación TIN2008-06566-C04-1.

Contents

I Introduction **1**

Introduction **5**

II Foundations **11**

1 Preliminaries **15**

1.1 Introduction 15

1.2 Introduction to Information Retrieval 16

1.2.1 Preprocessing and representing documents 17

1.2.2 Indexing: Inverted file 20

1.2.3 Representing information needs: queries 21

1.2.4 Information Retrieval models 22

1.2.4.1 Boolean model 23

1.2.4.2 Vector space model 23

1.2.4.3 Basic probabilistic model 24

1.2.5 Methods to improve Information Retrieval: Relevance Feedback 25

1.2.5.1 Relevance Feedback 26

1.2.6 Retrieval evaluation 27

1.2.6.1 Test collections 27

1.2.6.2 Efficiency evaluation in Information Retrieval . . 28

1.3 Introduction to structured Information Retrieval 30

1.3.1 Representing and preprocessing structured documents . . . 31

1.3.2 Indexing 33

1.3.3 Representing information needs: queries 35
1.3.3.1 Content-only queries 35
1.3.3.2 Content and Structure queries 35
1.3.4 Structured Information Retrieval models 36
1.3.4.1 Element scoring 37
1.3.4.2 Contextualization 38
1.3.4.3 Propagation 38
1.3.4.4 Aggregation 38
1.3.4.5 Merging 39
1.3.4.6 Processing structural constraints 39
1.3.5 Presenting results 40
1.3.6 Retrieval evaluation 41
1.3.6.1 Initiative for the Evaluation of XML retrieval . . 41
1.3.6.2 Evaluation measures in INEX 43

2 The context-based influence diagram approach for XML retrieval 49
2.1 Introduction 49
2.2 Preliminaries 51
2.2.1 Bayesian networks 51
2.2.2 Influence diagrams 53
2.3 Type of structured documents 55
2.4 The underlying Bayesian Network 55
2.4.1 Virtual nodes 59
2.5 The Influence Diagram model 60
2.6 Solving the Influence Diagram 63
2.7 Computing probabilities 63
2.7.1 Calculus of unidimensional posterior probabilities 64
2.7.2 Approximating the bi-dimensional posterior probabilities . 65
2.8 The Garnata information retrieval system for XML documents . . 66
2.8.1 Indexing subsystem 66
2.8.1.1 Application level 66
2.8.1.2 Physical level and data structures 67
2.8.2 Querying subsystem 68

III Methodological Contributions 71

3 Improving the context-based influence diagram model for XML retrieval 75
3.1 Introduction . . . 75
3.2 Adapting Garnata to the INEX tasks . . . 76
3.3 Parametrizing Garnata . . . 80
3.3.1 Recomputing weights depending on the type of unit . . . 80
3.3.2 Recomputing utilities depending on the type of unit . . . 82
3.3.3 Experimental evaluation . . . 84
3.4 Parametric utility model . . . 85
3.4.1 Experimental evaluation . . . 86
3.5 Concluding remarks . . . 93

4 Content and Structure queries 95
4.1 Introduction . . . 95
4.2 Introduction to structured queries . . . 96
4.3 Managing structured queries . . . 97
4.3.1 Combining probabilities . . . 102
4.3.2 Probabilities of disjunctions and conjunctions . . . 103
4.3.2.1 Noisy-OR gates . . . 103
4.3.2.2 Noisy-AND gates . . . 104
4.3.3 Implementation details . . . 104
4.4 Related work . . . 105
4.5 Experimental evaluation . . . 109
4.6 Concluding remarks . . . 119

5 Relevance Feedback for XML retrieval 121
5.1 Introduction . . . 121
5.2 Related work . . . 122
5.3 A proposal for content-oriented Relevance Feedback . . . 124
5.3.1 Generating the new query . . . 125
5.3.2 Computation of probabilities of relevance for original and expansion terms . . . 126

5.3.3 Inference and decision making with expanded queries . . . 128
5.4 A proposal for content-oriented Relevance Feedback of Content and Structure queries . . . 129
5.4.1 Relevance Feedback and Content and Structure queries . . 130
5.4.2 Selecting the expansion terms . . . 134
5.5 Experimental evaluation . . . 135
5.5.1 Data set and evaluation measures . . . 135
5.5.2 Evaluation methodology . . . 136
5.5.3 Experimental results for the Content-oriented approach . . 137
5.5.4 Experimental results for the Content and Structured-oriented approach . . . 144
5.6 Conclusions and future research . . . 150

IV Applications to the Andalusian Parliament 151

6 Description of the problem for the Andalusian Parliament 155
6.1 Introduction . . . 155
6.2 The case study: The Andalusian Parliament and its digital library 158
6.3 Detected weaknesses and possible improvements . . . 161
6.4 Methodology and system architecture . . . 163

7 XML collections of the Andalusian Parliament 169
7.1 Introduction . . . 169
7.2 Description of the problem . . . 171
7.3 Structure of the XML records of parliamentary proceedings files. Designing the DTD . . . 174
7.4 The conversion process . . . 178
7.4.1 General view of the process . . . 178
7.4.2 Text extraction . . . 180
7.4.3 Lexical analyzer . . . 181
7.4.4 Syntax analyzer and XML files creation . . . 185
7.5 Related work . . . 186
7.6 Some information on the XML collection . . . 188

7.7 Conclusions and future work 189

8 Managing videos: Segmentation, synchronization and retrieval 193
8.1 Introduction 193
8.2 Synchronization of video and records of parliamentary proceedings via XML files 196
8.3 Synchronization of video and records of parliamentary proceedings via video segmentation and annotation 200
8.3.1 Revision of existing annotation tools 200
8.3.2 Parliamentary solution 202
8.3.3 The segmentation algorithm 203
8.3.4 Optimization of the algorithm efficiency 205
8.3.5 Experimentation and evaluation 208
8.3.6 Features of the segmentation application 210
8.3.7 The synchronization tool 213
8.4 The video player 215
8.5 Conclusions and future work 216

9 A web interface for the Andalusian Parliament digital library 219
9.1 Introduction 219
9.2 Formulating queries 220
9.2.1 Description of the user interface for Content-only queries . 221
9.2.2 Description of the user interface for Content and Structure queries 223
9.2.2.1 From the visual query to the NEXI query 229
9.3 Presentation of the results 232
9.4 Interaction with the Relevance Feedback module 233
9.5 Conclusions and future research 235

V Conclusions 237

10 Conclusions and Future Works 241
10.1 List of publications 243

10.2 Future works 245

References **268**

List of Figures

1.1 Complete Information Retrieval process. 18
1.2 Single inverted file. (Stopwords have been deleted and stemming has been done). 20
1.3 An application of Rocchio's algorithm. Some documents have been labeled as relevant and non-relevant and the initial query vector is moved in response to this feedback. 27
1.4 Precision vs. Recall. 29
1.5 A fragment of an XML document. 32
1.6 The XML document from fig. 1.5 as a tree. 33
1.7 Example of XML document and its associated tree. 37

2.1 An example of a Bayesian Network. 52
2.2 An example of an Influence Diagram. 54
2.3 Example of the structure of a scientific article. 56
2.4 Bayesian Network representing a structured document collection. . 57
2.5 Fragment of Bayesian Network containing a virtual unit. 60
2.6 Topology of the Influence Diagram. 62
2.7 Relations between terms (circles) and units (squares). 68

3.1 Example of "focused task". 77
3.2 Example of "relevant in context" task (ranking for each criterion between brackets). 79
3.3 Centroids of the documents for the "best in context" task. 81
3.4 Function $x\frac{e^{x^n}-1}{e-1}$, for $n = 0, 1, 2, 3, 5$. 87
3.5 Results of the measures of retrieval effectiveness in the "focused" task, for $n = 0, 1, 2, 3, 5$. 88

3.6 Results of the measures of retrieval effectiveness in the "relevant in context" task, for $n = 0, 1, 2, 3, 5$. 89
3.7 Results of the measures of retrieval effectiveness in the "best in context" task, for $n = 0, 1, 2, 3, 5$. 90

4.1 Tree structure of the documents for Example 1. 98
4.2 Structural units in the hierarchy where some pieces of text have been specified for Example 1. 98
4.3 Architecture of a system for managing Content and Structure queries. 99
4.4 Recall-precision curves for the augmented CAS and the base-CO systems. 111
4.5 Recall-precision curves for the augmented CAS and the base-CAS systems. 113
4.6 Recall-precision curves for the augmented CAS and the SKR systems. 115

5.1 Candidate Terms. 125
5.2 Assessed elements with their corresponding set of generators. . . . 132
5.3 Improvement percentages using Relevance Feedback. 143
5.4 Differences in AiP, respect to non-Relevance Feedback case, when we use the best number of terms and the best fixed number of terms (6), for the case $m = 10$. 144
5.5 Differences in iP[0.01], respect to non-Relevance Feedback case, when we use the best number of terms and the best fixed number of terms (1), for the case $m = 5$. 145
5.6 Differences between AiP values, respect to non-RF case, when we use the best number of terms, for all the queries and $m = 10$. . . . 146

6.1 Architecture of *Seda*. 166

7.1 General information section. 173
7.2 General information section of a record of parliamentary proceedings in XML format. 174
7.3 Agenda of a record of parliamentary proceedings. 176
7.4 Summary of a record of parliamentary proceedings. 177
7.5 Development of a record of parliamentary proceedings. 178

7.6 Tags structure of a record of parliamentary proceedings. 179
7.7 Grammar corresponding to the agenda section. 186
7.8 DTD file of the official bulletins 190
7.9 Tags structure of a official bulletin. 191

8.1 Sets for synchronization 199
8.2 Example of synchronization 200
8.3 Differences between shot histograms 204
8.4 Comparison of non-contiguous shots and posterior refinement . . . 206
8.5 Shot with a reduction factor of 3. The lower left part is considered 207
8.6 Time vs. Reduction Factor 209
8.7 Accuracy vs. Time 210
8.8 Segmentation time for different numbers of shots ignored (first step) 211
8.9 Segmentation time for different numbers of shots ignored (second step) 211
8.10 Screen shoot of the Segmentation tool 213
8.11 Screen shoot of the Annotation tool 214
8.12 Screen shoot of the search result screen with the video player . . . 216

9.1 Form to introduce queries in the Andalusian Parliament webpage 221
9.2 Form to introduce Content-only queries in our search engine, *Seda* 222
9.3 User interface for Content and Structure queries in *Seda*. 227
9.4 Query elements in the DTD file of the records of parliamentary proceedings. 231
9.5 Presentation of results in *Seda*. 232
9.6 Visualization of the XML document in *Seda*. 233
9.7 Check buttons to indicate relevant information. 234
9.8 Relevant results selected by the user. 235

List of Tables

3.1 Importance of the different types of units used in the official runs. 83
3.2 Relative utility values of the different types of units used in the official runs. 84
3.3 Results for the "focused" task. 85
3.4 Results for the "relevant in context" task. 85
3.5 Results for the "best in context" task. 85
3.6 Relative positions in the INEX ranking for the different measures in the "focused task", for $n = 0, 1, 2, 3, 5$. 87
3.7 Relative positions in the INEX ranking for the different measures in the "relevant in context task", for $n = 0, 1, 2, 3, 5$. 87
3.8 Relative positions in the INEX ranking for the different measures in the "best in context task", for $n = 0, 1, 2, 3, 5$. 88
3.9 Runs submitted to the INEX 2008 *adhoc* tasks and positions in the rankings. (Foc: Focused, RIC: Relevant in context, BIC: Best in context). 91
3.10 Comparison between runs with and without applying the transformation in eq. 3.3. (Foc: Focused, RIC: Relevant in context, BIC: Best in context). 92
3.11 Runs retrieving only content-bearing elements and positions in the rankings. (Foc: Focused, RIC: Relevant in context, BIC: Best in context). 93

4.1 Comparison between the augmented CAS and the base-CO systems. 111
4.2 Comparison between the augmented CAS and the base-CO systems broken down by year. 112

4.3 Comparison between the augmented CAS and the base-CAS systems. 113
4.4 Comparison between the augmented CAS and the SKR systems. . 114
4.5 Results of the experiments with the *Wikipedia* collection using the PF/Tijah system. 116
4.6 Results of the experiments with the *IEEE* Computer Society collection using Garnata. 118
4.7 Results of the experiments with the *Wikipedia* collection using Garnata for different weights of the OR/AND gates. 119

5.1 Relevance Assumptions. 131
5.2 Query statistics. 136
5.3 Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement) 138
5.4 Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement) 138
5.5 Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement) 139
5.6 Results of the experiments in the following alternatives: Without Relevance Feedback, using m judged units and k expanded terms, and the best number of expanded terms for each query, with $m = 5, 10$. 140
5.7 Results of the experiments in the following alternatives: Without Relevance Feedback, using m judged units and k expanded terms, and the best number of expanded terms for each query, with $m = 20$. 140
5.8 Improvement percentages obtained comparing the results with Relevance Feedback to the results without Relevance Feedback, using $m = 5, 10$. 141
5.9 Improvement percentages obtained comparing the results with Relevance Feedback to the results without Relevance Feedback, using $m = 20$. 141
5.10 Number of queries where RF using the best number of expanded terms obtains worse or equal results than without Relevance Feedback. 146

5.11 Results of the experiments for the context expansion. 147
5.12 Results of the experiments full expansion and target expansion. . 147
5.13 Results of the experiments varying the number of expansion terms (k) in context subqueries and fixed to 4 in target subqueries. . . . 149

7.1 Information about digital library. 189

8.1 Percentage of correct segments detected (accuracy) with and without refinement. 208

1 List of the 34 INEX 2006 Content and Structure queries in NEXI format used in the experiments with the *Wikipedia* collection. . . 248
2 List of the 36 INEX 2007 Content nd Structure queries in NEXI format used in the experiments with the *Wikipedia* collection. . . 249
3 List of the 20 INEX 2008 Content and Structure queries in NEXI format used in the experiments with the *Wikipedia* collection. . . 250
4 Detailed results per query for iP[0.01] and AiP using the 34 INEX 2006 Content and Structure queries. 251
5 Detailed results per query for iP[0.01] and AiP using the 36 INEX 2007 Content and Structure queries. 252
6 Detailed results per query for iP[0.01] and AiP using the 20 INEX 2008 Content and Structure queries. 253
7 Number of times that the Augmented CAS is better/worse/equal that Base-CO and Base-CAS methods for iP[0.01] and AiP, using the 90 INEX-Wikipedia Content and Structure queries. 253
8 List of the 22 INEX 2003 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection. 254
9 List of the 19 INEX 2004 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection. 255
10 List of the 14 INEX 2005 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection. 255

Part I

Introduction

Introduction

Motivation

The increasing use of new technologies like computers and the Internet has supposed that most organizations which were used to work with documents in physical format, i.e., paper, have changed these trends, working with documents in electronic format nowadays. Due to these changes, several solutions have been proposed in the literature to give an easier access to this information and avoid the "information overload" problem. One of these solutions are Information Retrieval systems which permit us to gather the relevant information for the users' needs, represented by queries, from the collection of documents in electronic format efficiently. Information Retrieval has been an important discipline since the 40's, as Shingal explains in [94].

The methodology performed to retrieve these results depends on the type of Information Retrieval model used by the search engine. It has been a very attractive field for many researchers, as we can see in the notably amount of publications, specialized journals and conferences, due to the wide range of tasks to be developed and the different types of alternatives that can be implemented in each one.

An important matter of Information Retrieval is the nature of the documents of the collection. Most of them are not flat documents because they contain a structure which can be very useful in retrieval tasks. A relatively new field, structured Information Retrieval, introduced by Chiaramella in [16], uses this organization of the documents to retrieve only the parts of them related to the users' needs instead of the whole ones. It could be specially beneficial when tackling with large documents and also with heterogeneous ones.

An example of large documents could be this dissertation, which has more than two hundred pages. In this sense, if the user is interested in the definition of Bayesian Network, a traditional Information Retrieval system will consider the whole document as relevant, whereas a structured Information Retrieval system will give as output the section 2.2.1 of chapter 2. Another example, which is the base that maintains our research, is the collection of documents of the Andalusian Parliament[1] due to the fort they deal with several different matters as for example policy, economy, agriculture, etc. In concrete, it contains the official documents (official bulletins and records of parliamentary proceedings) and session videos from the sixth to the eighth legislature (about 12 years) dealing with the information produced in the Parliament such as law initiatives, comissions, etc.

The citizens can request information in natural language format (queries) about different topics like "laws about education in Granada", "income tax in Andalusia" and "agricultural initiatives in Almeria". Then, the Information Retrieval system will give as output the parts of the documents (and the related videos) which are relevant.

To manage structured information, it is necessary to use markup languages, like XML, since they can reflect both the content and organization (structure) of the documents. Apart from the representation of the documents, the queries can introduce some structural restrictions using this representation too, transforming them into more specific requests.

Our purpose is to develop an structured Information Retrieval system that might be used by the Andalusian citizens. So, another important aspect is the development of a web interface to interconnect the users with the Parliament collection easily, being really necessary in the case of structured queries because users do not exactly know how to create them in a proper way. Also, the results from a query can be shown using different strategies which can be adapted to the users' needs and the intregation of the Relevance Feedback framework in the system to get results more useful for users' requests.

Finally, although our motivation is Andalusian Parliament, our approach must be able to deal with any type of documents, apart from the documents of the

[1] http://www.parlamentodeandalucia.es

Andalusian Parliament collection. In our case, we have used the test collection from the INEX workshop[1] containing a document collection from *Wikipedia*.

Main contributions of the dissertation

The first contribution of the dissertation has been the different improvements developed in the base system, Garnata, created by members of the department of Computer Science and Artificial Intelligence (DECSAI[2]) of the University of Granada.

In concrete, the changes are performed in the context-based influence diagram model, described by Campos et al. in [22], implemented in Garnata. Firstly, we have to incorporate new features in order to adapt the results retrieved by the system to several retrieval tasks proposed in the INEX Workshop, which fit perfectly into real use-cases. The tasks are *focused*, *best in context* and *relevant in context* (see section 1.3.5).

By means of this new feature, we can participate and compare the results given by our system with the results of other systems in INEX 2007 and 2008. Then, these changes permit us to know the performance of the system. Apart from that, these tasks have been used in the Andalusian Parliament system as different stategies to present results to the users. It is an important improvement because it is not very common to find search engines which have the option of retrieving their results organized following different strategies, being a help to the users of the Andalusian Parliament system.

There are several aspects in the Information Retrieval field such as the size of the structural units, the types of units to retrieve, or even the number of query terms in the retrieved units which are highly important in the structured Information Retrieval process. In our research, these aspects have been modeled by means of a set of parameters representing in general way an utility criterion. Therefore, the process of estimating these parameters has been modified to obtain a better effectiveness in the structured Information Retrieval process.

[1] http://www.inex.otago.ac.nz/
[2] http://decsai.ugr.es/

To describe this contribution in more detail, the original model implemented in Garnata considered that all the structural units had the same importance, so it retrieved all types of units but it is not useful because there are several of them which are not interesting from a retrieval point of view. Therefore, the model was changed to use two different parameters (weight and utility) to control this aspect. Following with this contribution, we last present an extension of the context-based influence diagram model based on the use of a parametric non-linear utility model which can penalize a unit if it does not contain all the query terms.

Another important contribution of the dissertation is the use of Content and Structure queries in our Information Retrieval system, Garnata. At first, it was designed to run Content-only queries in natural language format retrieving any type of structural units. As we are working with a structured system, it was important to develop a methodology which permits us to introduce more specific queries where the user could indicate their preferences in both content and context. The user only gets the types of units indicated by the structural restrictions satisfying the content requirements too.

There exist several works in the literature dealing with Relevance Feedback, however the amount of publications decreases when we refer to Relevance Feedback in structured Information Retrieval. This introduction gives an idea of our following contribution focused on the creation of a Relevance Feedback framework to analyze the information given by the users to the retrieved elements. This information consists of the relevance assessments of the users for some units retrieved by the system. Using this information, a new query is created which is more adapted to the users' requests. It is important to mention that our Relevance Feedback framework can be used for both types of queries (Content-only and Content and Structure queries).

Finally, *Seda* is the last contribution of the dissertation, which consists of a web interface which permits the users to interact with Garnata using the collection of official documents and videos of the Andalusian Parliament. *Seda* facilitates the Information Retrieval process to the users who can not know anything about the collection or Information Retrieval. As a result of a query in *Seda*, the user

gets all the relevant units to the query, together with the video streams associated to these units.

Before using any of these contributions in Garnata with the Parliament collection, the system was checked using the INEX collection commented in the previous section. The results of these experiments indicated a good performance of the system.

Chapter overview

The dissertation is organized in five different parts. Firstly, Part I consists of this preface to introduce the dissertation.

Part II contains two chapters with the foundations to understand different concepts of Information Retrieval, and the base structured Information Retrieval system, Garnata, is explained in more detail.

Specifically, the chapter 1 introduces the main concepts about both traditional and structured Information Retrieval. This chapter describes the indexing problem, several types of Information Retrieval models, the different tasks to present results to the users, Relevance Feedback and the retrieval evaluation.

In order to understand the base system of the dissertation, Garnata, the chapter 2 focuses on showing the context-based influence diagram model implemented in it. It is necessary to know the methodology of the model to understand several contributions of the dissertation because they are directly developed in the model.

Part III contains the main methodological contributions of this dissertation, presented on previous section. Thus, chapter 3 describes all the improvements developed in Garnata: The different ways to present results based on the tasks of INEX 2007 and 2008, the use of the weight and utility parameters and the development of the parametric non-linear utility model. In chapter 4, we can see the methodology to incorporate Content and Structure queries in Garnata. Lastly, chapter 5 describes the Relevance Feedback framework developed in Garnata for both types of queries.

After seeing these contributions, the application part of the dissertation, which is another contribution of the dissertation, is shown in Part IV. Firstly, chapter 6

describes the problem of the Andalusian Parliament detecting its weaknesses and possible improvements. Then, we talk about the collections of the Andalusian Parliament showing its structure and the conversion process from PDF to XML format in chapter 7.

As we commented before, the collection of the Andalusian Parliament consists of documents and videos and both sources of information are integrated. Therefore, it is necessary to have different mechanisms to segment, synchronize and retrieve the videos with the document units which are shown in chapter 8. Lastly, the chapter 9 presents the web interface for the Andalusian Parliament digital libray, *Seda*, which is in charge of interacting with Garnata. In addition, the interface is very intuitive for the users of the system because they do not know how it requires the inputs. Then, the communication process transforms the information from the users into a format used by Garnata and vice-versa.

Finally, Part V consists of a chapter where all the conclusions and future works of the dissertation are stated. The list of publications supporting the contributions and applications of this thesis is included, as well.

Part II

Foundations

Chapter 1

Preliminaries

1.1 Introduction

Information Retrieval (IR) is a discipline that, in its beginning, considered documents as a whole, i.e., a document was a set of terms describing its content. In fact, all the stages of the traditional IR systems work with flat documents. For instance, the indexing process takes all the terms from a document, after preprocessing it to get the terms that best summarize it, without any difference among them. Then, the retrieval process retrieves the relevant documents for a given query.

However, documents such as textbooks, scientific articles, technical manuals, etc. have two main characteristics: on the one hand, the set of terms used to describe their contents, and on the other, a well-defined structure to organize these contents intelligibly improving readability for the user, as chapters, sections, paragraphs, etc. Since both characteristics are quite important and must be taken into account when writing a document, the basic idea of the traditional IR was improved to consider the internal organization of the documents too.

As a consequence, all the processes related to the traditional IR field had to be adapted to manage this type of documents, called *structured documents*, incorporating this additional feature of them in the different stages. So, retrieving information from structured documents differs from retrieving information from flat documents. This type of IR receives a new name, structured IR.

In this chapter, we shall review those main elements in IR and how each one is influenced by the fact of using structured documents. So, this chapter proposes a general view of all the concepts, definitions, and important aspects on IR needed to understand the dissertation. Thus, we shall organize the chapter as follows: Firstly, we shall describe the traditional IR process (section 1.2) in detail: *documents* and the way to represent and preprocess them in section 1.2.1, the *indexing process* in section 1.2.2, the way to represent the users' information needs (*queries*) in section 1.2.3, different traditional IR *models* in section 1.2.4, an introduction to Relevance Feedback in section 1.2.5 and the evaluation of the retrieval process in section 1.2.6.

Afterwards, we shall introduce structured IR (in concrete, XML IR) in section 1.3. XML markup language and its main features are commented in section 1.3.1, the way to index the XML documents in section 1.3.2, queries for XML IR in section 1.3.3, the taxonomy of other structured IR models in section 1.3.4, the different methods to present results in section 1.3.5 and the evaluation of this type of IR systems in section 1.3.6.

1.2 Introduction to Information Retrieval

IR has been an important discipline since the 40's due to the increase of the number of documents (for example, scientific papers) that needed to be easily accessed. Then, the meaning of the term IR can be very broad because it is used in a lot of different contexts where the information needs to be represented, stored, organized and accessed corresponding to the IR tasks. However, in our context, IR might be defined, according to Manning et al. in [65], as:

> IR is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

At the beginning, IR was not a popular activity because there were few people interested on it: reference librarians, paralegals, and similar professional searchers. But, this trend changed over the years and nowadays it is really common using any type of IR system like a search engine or an email searcher. In

conclusion, IR is the most dominant form of information access, overtaking traditional database style searching.

After defining the concept of IR, we shall introduce the different components of the IR systems. To a better understanding of all of them, the whole IR process is shown in fig. 1.1: *Documents* are the source of information where the users can find whatever they are interested in. They must be transformed to be interpreted and easily accessed by the IR system obtaining the set of key words for each document which is known as the *indexing process*. After the document collection has been created, it is prepared to be easily accessed by the system introducing *queries* in the IR system. The IR system runs its search engine and compares each document with the query, obtaining, in some cases, the similarity degree of each document to the query or only those documents that satisfy the query absolutely in other cases.

Every search engine has its own methodology to retrieve the set of documents (results) for each query, which is called *IR model*. The next step is to present the results of the search engine to the users. These documents might be evaluated by themselves deciding if the results are satisfactory or not to their information needs. This evaluation could be used to generate a new query, thus the search engine is run again. This process is called *Relevance Feedback*. In experimental environments, the IR systems include an additional module to evaluate the quality of the results (*performance evaluation*).

1.2.1 Preprocessing and representing documents

In this section, we shall describe the first entity of an IR system, the document, and the way it is represented and processed. Thus, a (text) *document* is a succession of words with some punctuation. Note that we identify a document with its textual content, and not with the physical one.

The representation of a document in a computer is an important task because a flat document does not permit an efficient management, being important to find an alternative representation of documents. Therefore, it is necessary a representation which facilitates the search process, reducing the size of the collection and giving an uniform appearance to it. It consists of extracting or identifying

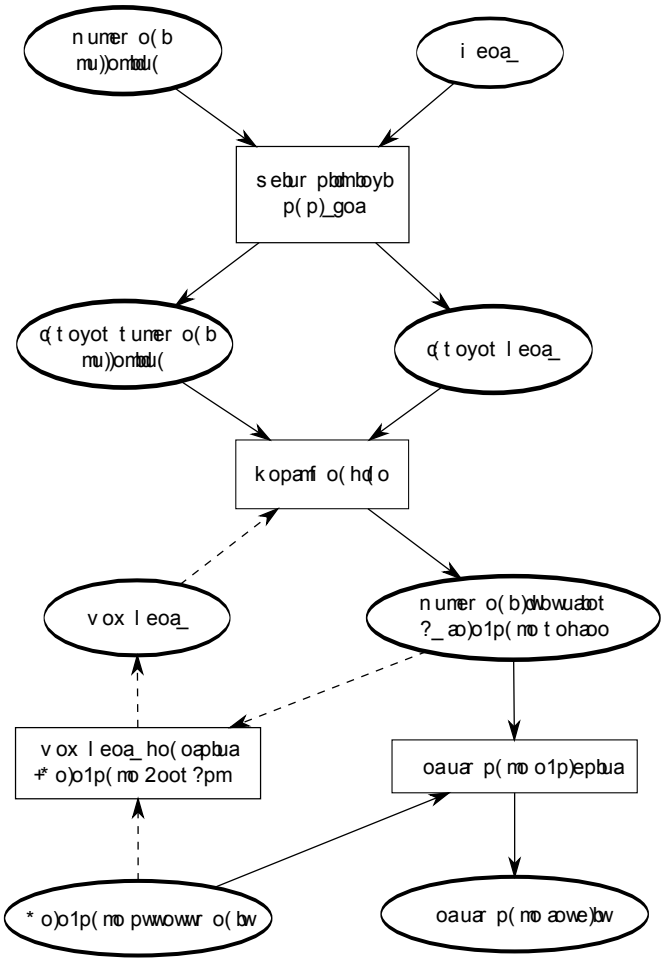


Figure 1.1: Complete Information Retrieval process.

from the documents the *key words* or *indexing terms*: They are a set of words that summarize the content of them.

This technique has three important advantages: on one hand, the necessary space in the hard drive to store the collections is lower; on the other hand, Baeza and Ribeiro et al. explain in [8] that we can avoid to store many words which do not have any importance from a semantic point of view and can introduce noise for future tasks. Lastly, this representation facilitates the work to computers, since they can use documents in an efficient way. The process of representing the content of the documents is called *automatic text analysis*.

Before introducing the terms in the analyzer, there are several actions which can be taken over the terms:

- *Tokenization*: It breaks up the document text into words called tokens.
- *Non alphabetic signs deletion*: punctuation marks, blank spaces, numbers (not always), etc.
- *Case folding*: It normalizes words transforming all the letters into their lower case version. Moreover, it usually deletes all the accents from the letters.
- *Stopwords deletion*: The system deletes all the words that are not important from a semantic point of view, like prepositions, articles, etc. Besides, Fernández in [31] and Fowler et al. in [33] show an approach where the most common and the least frequent words are deleted too, because they do not offer important information about the document in which they appear. For every language, there exist complete *stopword lists*, so it is not necessary to create them. The most common stopword list is the used one in the SMART system as Rocchio describes in [78].
- *Stemming*: For every term, the system only takes its root. Therefore, the main idea of this action is to consider as synonyms all the terms with the same root. Following this idea, the system transform plural words and different verb compositions into their roots, keeping only the core information. It is not an easy problem and depends on the language we are using. There

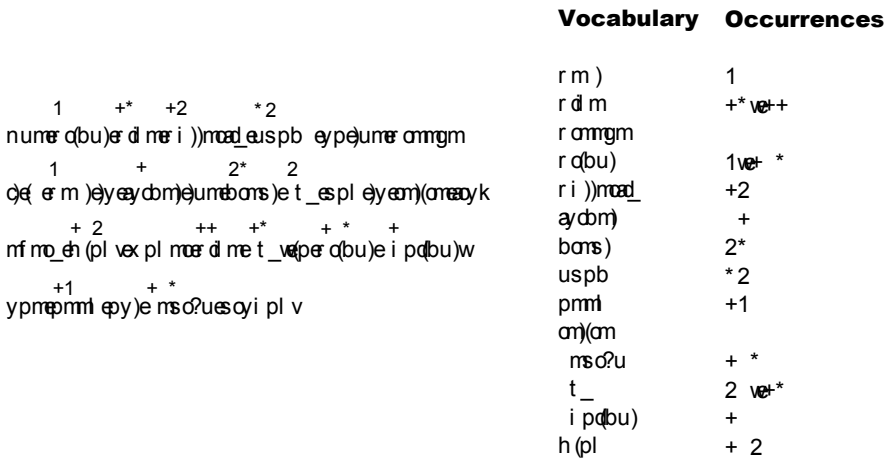


Figure 1.2: Single inverted file. (Stopwords have been deleted and stemming has been done).

are several algorithms to stem words but the main one is the *Porter stemmer*. In [51], Hull does a comparison of these algorithms. This action is optional.

1.2.2 Indexing: Inverted file

The index structure allows the system to access efficiently to the logic view of the document presented in the previous section. As an example of an index structure, we are going to review *inverted files*. This is one of the most important advances in the history of IR systems.

An inverted file is a mechanism to index a document, generally text. It consists of two parts: *vocabulary* and *occurrences*. The vocabulary is the set of different words appearing in the text, and occurrences indicate the places (document identifiers, and sometimes, positions inside those documents) where these words appear in the text. (we can see an example in fig. 1.2).

The required space for the vocabulary is small, however the occurrences need more space because each word appearing in the text is referred once in the structure. Even if the stopwords are deleted from the index, the table of occurrences can get a size between 30% and 40% of the size of the original text. So, vocabulary and occurrences are usually stored in different files. In many cases, this

vocabulary can be stored in the main memory. For big collections, the vocabulary size is not more than 40 MB, as Witten et al. explain in [104].

1.2.3 Representing information needs: queries

A query language may be used at the querying stage to formulate requests by the users. Natural language is the most common way to represent the queries in the traditional IR systems, for instance "laws about agriculture in Andalusia".

Once the query has been created and introduced in the system, it is preprocessed like the documents of the collection to maintain the uniformity. Then, the preprocessed query is searched in the inverted file which is a really fast process. It consists of the following three steps:

1. *Vocabulary search*: Words or patterns found in the query are separated and placed in the vocabulary.

2. *Occurrences retrieval*: The list of occurrences of these words is extracted from the system.

3. *Occurrences handling*: The lists of occurrences, obtained in the previous step, are processed to find the correspondence of the documents with the query terms.

The *unary queries* can be solved using any method which increases speed, like *hashing* or *B-trees*, as Aho et al. describe in [2]. If the query is composed of several words, it is necessary to manipulate the different lists of occurrences.

Queries based on context or proximity are much more difficult to solve using inverted files. Each element must be searched separately, generating a list for each one. Then, the set of lists is run synchronously, searching the places where the words from several lists in sequence appear (for sentences) or are very close (for proximity).

1.2.4 Information Retrieval models

An important concept in IR is the *retrieval model*. It is defined as a specification of the documents and query representation and the way to compare each other to retrieve the relevant documents. Baeza, Ribeiro et al. [8] present a formal characterization of this concept in the following way:

- $\mathcal{D}$ is the set of representations of the documents from the collection.
- $\mathcal{Q}$ is the set of representations of the users' information needs.
- $\mathcal{F}$ is a frame to model representations of documents, queries and their relationships.
- $Sim(D_j, Q)$ is a function to associate a real number to the pair (D_j, Q), $Q \in \mathcal{Q}$ and $D_j \in \mathcal{D}$. This value permits us to give a ranking of documents with respect to the query Q.

Directly related to the function Sim, one of the most important aspects of IR is the concept of *relevance*, defined by Saracevic in [83], which was born in the 40's like this discipline, although it is not completely understood yet. It is usually considered as a binary concept taking one of the following values: *relevant* and *non-relevant*. Therefore, it is a subjective concept depending on the decision of the user, as Bookstein explain in [9].

Then, the objective of the function Sim is to determine the relevance degree of a document according to a query. This task was introduced in the IR area by Maron and Kuhn [66], where they gave a probabilistic aspect to that function, measuring the probability of the similarities between a document and a query $Sim(D_j, Q)$ which determined the acceptance of the document by the user.

Let T be the number of terms in the system and t_i a generic term. Let $\mathcal{T} = \{t_1, ..., t_T\}$ be the whole set of terms. A weight $w_{i,j} > 0$ is associated to every term t_i of a document d_j. Generally, the bigger the weight $w_{i,j}$ is, the more important the term t_i is inside the document d_j. On the other hand, if a term t_i is not contained in the document d_j, the weight $w_{i,j} = 0$. Besides, the document d_j has assigned a weight vector $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$. Finally, g_i is defined as

a function, applied to the vector $\vec{d_j}$, which returns the weight associated to the $i-th$ term of the vector $\vec{d_j}$, i.e. $g_i(\vec{d_j}) = w_{i,j}$.

It is usually assumed that weights of the different terms are independent, i.e. knowing a weight $w_{i,j}$ associated with a pair (t_i, d_j) does not give information about the weight $w_{k,j}$, associated with the pair (t_k, d_j) where $k \neq i$. This is a simplification because the occurrences of a term in a document are not independent.

Thus, an IR model will be completely defined computing the function $Sim(D_j, Q)$, combining in some way these weights. Afterwards, we shall introduce the three classic IR models:

1.2.4.1 Boolean model

The Boolean model was the first model in the literature and has been very used in IR systems. Its framework is composed of the documents represented as sets, the queries represented as boolean expressions and the different opperations to work with sets (union, intersection and complementary set, described in more detail in [8]). The formulation is very intuitive:

A query composed of boolean expressions of terms is introduced in the system. Then, the boolean expression is evaluated using the previous opperations with the sets of documents where every query term is contained. The result is the set of documents that verify the boolean expression following a binary decision test, i.e., a document is relevant or not for a query (similarity is one or zero, respectively).

Its implementation is really simple, but this model has several disadvantages like this binary decision test does not distinguish between documents, so all the relevant documents have the same relevance degree. Then, the list of results can be very disordered with respect to the *real* relevance of the documents.

1.2.4.2 Vector space model

This model, presented by Salton et al. in [82], considers every document as a *vector* in the space of the set of all the vocabulary terms. In addition, this model permits a representation of the query following the same format of the document.

The *distance* (similarity) between a document and the query is defined as the cosine of the angle among the two vectors[1].

Then, the system implementing this method is really straightforward: given a query, it is transformed into a vector and the distances between this vector and the document vectors are computed. The greater the similarity is, the more relevant documents are for the query.

The disadvantage of this model is that it penalizes large documents with a lot of terms.

1.2.4.3 Basic probabilistic model

This model was defined by Robertson and Sparck-Jones [77] and later C. J. van Rijsbergen [98] describes it as well. It is based on the *probability ranking principle* which assures that the best retrieval performance is achieved when the documents are sorted according to their probabilities to be judged as relevant with respect to a query. These probabilities are computed in the most accurate way using the available information.

Thus, given a query q and a document d_j of the collection, the model tries to estimate the probability that the user finds this document as relevant. This model assumes this probability only depends on the query q and the document representation d_j in the system.

Once we know the probabilities of the document to be relevant and non-relevant for the query, the similarity function is based on them to get a relevance value of the document. In this case, this similarity function is computed as the division of the probability of the document to be relevant with respect to the probability of the document to be non-relevant.

This expression can be derived (using the Bayes theorem and supposing the indexing terms are independent) to obtain a simple formula corresponding to the valuation function in the probabilistic model.

The main advantage of the probabilistic model is that documents are sorted in decreasing order of their relevance probabilities. Disadvantages are: we need to estimate the initial separation between the relevant and non relevant sets

[1] The cosine of the angle is computed as the scalar product between two normalized vectors.

of documents, the weights are binary and the assumption that the terms are independent, although this assumption is practically necessary to try to solve this problem.

1.2.5 Methods to improve Information Retrieval: Relevance Feedback

A user does not usually know how to express in a proper way an information need, so the set of selected terms has to be refined until a "good" query is formulated, for example, users might use the term *plane* whereas the document in the collection might use *aircraft*.

Therefore, the users usually solve this problem refining the query manually, but there exist several methodologies which can help to perform the query refinement, either fully automatically or with the user's support.

In [65], Mannig et al. describe these methodologies in detail. They are classified in two major classes: *global* and *local methods*. Global methods consists of techniques for expanding or reformulating query terms, independently of the query and the results returned from it. Then, changes in the query are based on the use of semantically similar terms. The global methods are:

- Query expansion/reformulation with a thesaurus.
- Query expansion via automatic thesaurus generation.
- Techniques like spelling correction.

Local methods adjust a query relative to the documents that initially appear to match the query. The basic local methods are:

- Relevance Feedback (RF).
- Pseudo RF, also known as Blind RF.

In this dissertation, we shall focus on RF, which is one of the most used and successful approaches.

1.2.5.1 Relevance Feedback

Relevance Feedback is a methodology based on the use of the relevance information given by the user for the results of the original query (feedback) to create a new query more adapted to the user's need. Then, the procedure is described in the following steps:

1. User introduces the original query.
2. IR system retrieves the results for the given query.
3. User judges a set of results as relevant or non-relevant giving feedback information.
4. A new query is generated using the feedback information of the previous step and the original query.
5. The system retrieves the set of results for the new query which must fulfill in a higher level the user's need.

It is really common that users do not know the collection well, so it is, in fact, hard for them to formulate a good query. Nevertheless, it is easier for the users to judge which documents are relevant or non-relevant in their search process. Then, RF can go through one or more iterations of the procedure, refining more and more the query in each one, until the users think they have found the information they were seeking.

The classic algorithm of the RF methodology is the Rocchio algorithm. It incorporates the RF methodology into the vector space model (see section 1.2.4.2) as we can see in [65].

The main idea is to find a query vector that maximizes the distances (high similarity) with relevant documents and minimizes the distances (low similarity) with the non-relevant documents. Then, the optimal query is the vector difference between the centroids of the relevant and non-relevant documents.

However, there exists a problem with this methodology because the full set of relevant documents is not known: it is what we want to find. Instead, an alternative was created based on the sets of known relevant and non-relevant

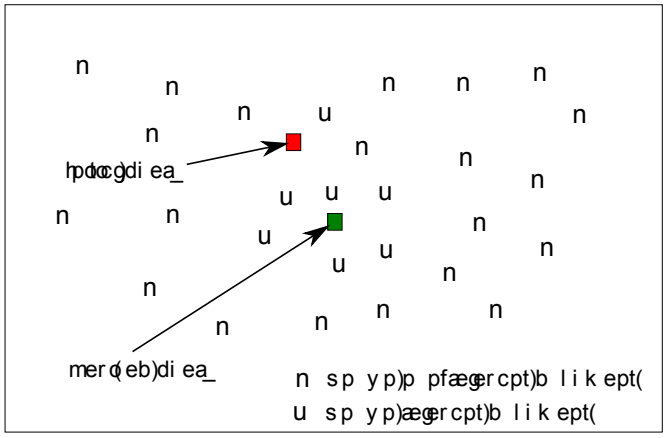


Figure 1.3: An application of Rocchio's algorithm. Some documents have been labeled as relevant and non-relevant and the initial query vector is moved in response to this feedback.

documents given by the user's feedback, so the bigger are the sets the more precise will be the methodology. We can see an example in fig. 1.3.

RF is a good methodology to improve recall and precision, but it is more useful to increase recall in situations which require it. This conclusion is due to the methodology that expands the query and the effect of the use case: if users want high recall, they take their time to review results and iterate on the search.

1.2.6 Retrieval evaluation

To measure the performance of IR models, the general evaluation process of effectiveness of an IR method consists of: given a set of retrieved documents of a query (obtained by the method) and a set of documents classified as the relevant ones for the query (by a group of experts), we want to measure the similarity between them. Then, an estimation of the goodness of this method is obtained.

1.2.6.1 Test collections

Before introducing the evaluation measures, it is important to note that to evaluate and study different IR models is necessary a standard test collection. The objective is to compare the result sets of one model with respect to other ones

given a set of queries and check the accuracy of all these models comparing their results with the relevance assessments. Then, a test collection is composed of three parts:

1. *A set of documents.*
2. *A set of queries.*
3. *Relevance assessments.*

There exist different test collections, for instance, **CACM** and **CISI** collections[1] among other. Nevertheless, **TREC (TExt Retrieval Conference)** collection[2] is currently the state of the art and the most used one in traditional IR.

1.2.6.2 Efficiency evaluation in Information Retrieval

Due to the fact there are several IR tasks, there exist different types of evaluation, described in [8], which are specific for every method too. From here on, we shall focus on the evaluation methods to indicate how accurate is the IR system retrieving relevant documents.

Precision and recall To evaluate these methods using *precision* and *recall* measures, we first note the set of relevant documents to the query as C_r[3] and the set of retrieved documents, defined in [81], obtained by the IR method as C. We define *precision* and *recall* as:

- *Precision* is the relevant subset from the set of retrieved documents:

$$\text{Precision} = \frac{|C_r \cap C|}{|C|}.$$

- *Recall* is the retrieved subset from the set of relevant documents:

[1] available in `ftp://ftp.cs.cornell.edu/pub/smart/cacm/` and `ftp://ftp.cs.cornell.edu/pub/smart/cisi/`, respectively

[2] availabe in `http://trec.nist.gov/data/test_coll.html`

[3] The cardinality of the A set is noted as $|A|$

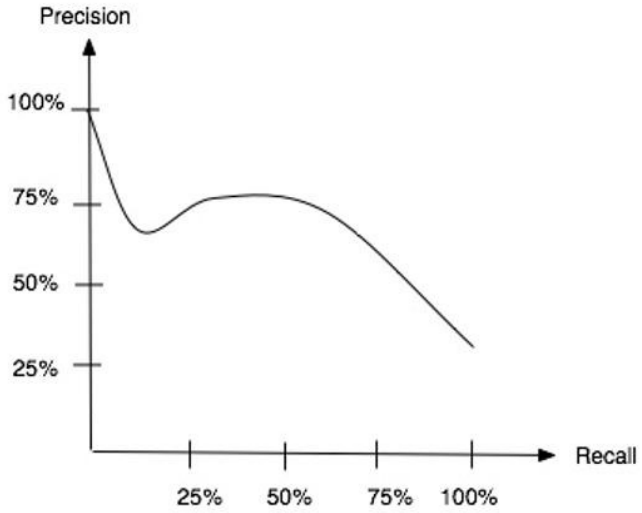


Figure 1.4: Precision vs. Recall.

$$\text{Recall} = \frac{|C_r \cap C|}{|C_r|}.$$

The higher both measures are, the better the algorithm. In these definitions, it is supposed that the user can access all the retrieved documents, but in fact, it does not usually happen. The user only reads some of the first retrieved documents. To improve this evaluation process, we obtain a graphical representation which shows a curve indicating precision versus recall. This curve is obtained computing the precision to different recall levels. For example, we can choose different recall levels like 10%, 20%,..., 100% and compute the precision for these ones. These precision values indicate the proportion of relevant documents for each given percentage.

As we can see in fig. 1.4, the graphical representation shows precision in the Y axis and recall in the X axis. Then, an IR method with higher curve[1] than another means the first method is better.

Each representation is for one query, so if we want to evaluate several queries, we would have a curve for each query. Thus, we define a new measure: *Average*

[1]With higher precision values to the same recall

precision in each recall level, which is computed as follows:

$$\overline{P(rl)} = \frac{1}{n} \sum_{i=1}^{n} P_i(rl),$$

where $\overline{P(rl)}$ is the average precision of the recall level rl, n is the number of queries and $P_i(rl)$ is the precision of the recall level rl for the query i. This measure is used to compare different IR methods graphically.

To study with more detail the set of queries and detect strange behaviors in some of them, there exist other additional measures described in [8], like average precision of relevant documents, R-precision or precision histogram.

1.3 Introduction to structured Information Retrieval

This specialized discipline is an extension of the traditional IR taking into account the context information, so we must know in detail all the aspects of traditional IR commented in the previous section to be able to understand the features of the structured IR.

The structured IR, used in this context as a synonym to XML IR, focuses on exploiting the available structural information in documents to implement a more focused retrieval strategy and return document components, the so-called XML elements, instead of complete documents in response to a user's query.

We shall only focus at one standard for encoding structured documents: eXtensible Markup Language or XML, which is currently the most used one. It is a standard for data representation and exchange on the Internet. It is expected to become an universal format for data exhange on the Web and thus in the future we shall probably find many documents in XML format on the Web. Nevertheless, most of what we say in this section is applicable to markup languages in general.

In XML, each document component is defined by a tag that is contained in the XML document with the content of the document. The organization of these tags in a document corresponds to the structure of it which follows a hierarchical

structure. Therefore, XML allows reflecting both the content and structure of a structured document.

We must distinguish between the two views of the XML content, as it is explained in [65]: data-centric and text-centric XML. On the one hand, a data-centric XML document is seen as a container for data, usually non-text data (worker profiles, invoices, flight schedules, etc). The document meaning depends on the structured data contained, and presents a regular and complex structure and homogeneous content. When the user formulates queries for such a type of XML document, he or she is interested in an exact match, such as a database-type style. On the other hand, text-centric XML documents usually represent text documents (books, e-mails, etc.), and the structure is more irregular and the data heterogeneous. The queries, although taking the structure into account, focus on the text and require ranking. The aim of text-centric XML retrieval is therefore to develop methods to find correspondence between the text of the query and the text of the XML documents (but also considering structural restrictions). Text-centric XML retrieval is the context in which this dissertation is set.

1.3.1 Representing and preprocessing structured documents

An XML document can be represented as an ordered, labeled tree where each node of the tree is an *XML element* corresponding to a tag of the XML document. With XML, we can create all the tags we need for representing documents. For each one, we need an opening and closing *tag* which indicate the boundaries of the XML element, i.e., all the text inside the tag represents the own text of the element and the other elements inside the tag are descendants of this one, so an XML document has a hierarchical structure. Another feature of an XML element is that it can have one or more *XML attributes* which are added in the opening tag. In the XML document in fig. 1.5, the *section* element is enclosed by the opening and closing tags `<section ...>` and `</section>`. It has an attribute *number* with value *1* and two child *subsection* elements.

Fig. 1.6 shows fig. 1.5 as a tree. The leaf nodes of the tree (*subsection* elements) consist of text, e.g., *XML is an area...* and *If the result of...* The tree

```
<article>
  <author>Carlos Martin</author>
  <title>XML IR</title>
  <chapter number=‘‘1’’>
    <section number=‘‘1’’>
      <subsection>XML is an area...</subsection>
      <subsection>If the result of...</subsection>
    </section>
  </chapter>
</article>
```

Figure 1.5: A fragment of an XML document.

internal nodes encode either the structure of the document (*title*, *chapter*, *section* and *subsection*) or metadata information (*author*).

XPath (*XML Path Language*) [1] is a language to locate and process any XML element from an XML document. It represents the path of an XML element following the tree structure indicating for each element in the path its name (tag) and index. An example of an XPath of the fig. 1.5 is `/article[1]/chapter[1]/section[1]/subsection[2]`, representing the second *subsection* element with the text *If the result of...* We shall also refer to paths as *XML contexts* or simply *contexts* in this section.

Another interesting concept in XML is the *schema* which describes the structure of an XML document indicating the constraints of an allowable document for a specific application. In fig. 1.5, a schema for *articles* may stipulate that *section* can only occur as children of *chapter* and that only *chapter* and *section* have the *number* attribute. Two standards for schemas for XML documents are *XML DTD* (document type definition), used in the dissertation, and *XML Schema*.

As in the case of traditional IR, representing the content of XML documents involves several preprocessing tasks: tokenization, stopword removal and stemming (see section 1.2.1).

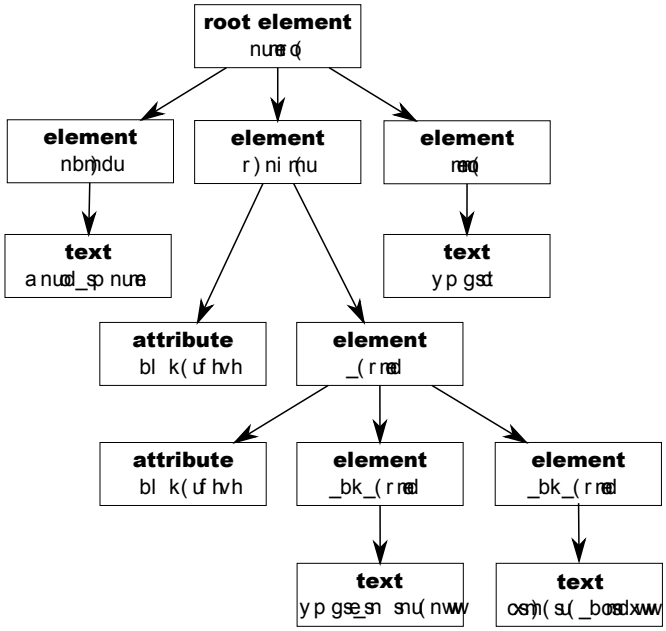


Figure 1.6: The XML document from fig. 1.5 as a tree.

1.3.2 Indexing

In XML retrieval, the retrieval units are not predefined (for instance, a whole document, a section or a paragraph can be potential answers to a query). Then, determining the elements to be indexed is an important matter because indexing all of them could not be necessary and efficient. Therefore, there exist different indexing strategies for this type of documents.

As we mentioned in the previous section, documents need some preprocessing taks which are performed in XML documents too, but they involve an additional step: parsing of the XML format.

In conclusion, this section presents different indexing strategies for XML retrieval describing the elements to index. All these strategies are described in more detail by Lalmas in [60], but we think it is interesting to introduce them in this section.

- *Element-based indexing*: The element-based indexing is the simplest approach because it indexes all the elements from all the levels of granularity. Here, each element is indexed based on the text contained in it and all

its descendants. For example, to index a section which contains two paragraphs, we extract the text from the section and the text from these two paragraphs.

- *Leaf-only indexing*: The leaf-only indexing strategy manages a index much smaller than the previous one, because it only indexes the leaf elements. Then, this index only permits us to estimate the relevance of these elements, so it is necessary to have a propagation mechanism to compute the relevance of the ancestors (non-leaf elements) based on the combination of the retrieval scores of their children. This mechanism works from the leaf elements to the root element following all the hierarchical structure.

- *Aggregation-based indexing*: The aggregation-based indexing strategy aggregates for each element the term statistics of its own text with the statistics of all its descendants. Then, this aggregated representation of the element is used to estimate the element relevance. The process is based on considering only leaf elements containing query words and their ancestors, at query time, involving the use of a small index as only the leaf nodes are indexed like in the previous strategy.

- *Selective indexing*: The selective indexing approach considers only a subset of elements as retrieval units. Therefore, it indexes these units reducing the size of the index although it is not as small as the index of the leaf-only indexing strategy. This strategy is mostly used in combination with another strategy which is usually in charge of computing the term statistics.

- *Distributed indexing*: The distributed indexing strategy creates an index for each element type which are more uniform in terms of vocabulary and size than the index built for all the elements of the collection. Then, the term statistics are computed separately for each index being more consistent due to their uniformity. When a new query is introduced, each index generates its own list of results and finally, all these lists are merged to create a single list of results.

- *Structure indexing*: The structure indexing strategy distinguishes different "structural contexts" of a term when computing term statistics. There are different alternatives for the structural contexts. One approach is based on computing the statistics for structure/term pairs and another approach is to associate a weight to a structural constraint to reflect its importance.

We are specially interested in the leaf-only indexing strategy because it is performed by the search engine used in the dissertation.

1.3.3 Representing information needs: queries

In structured IR, we can find two different ways to represent the users' information needs:

1.3.3.1 Content-only queries

A content-only query (CO) is a non-structured query in natural language. This type of query is similar to the queries of the traditional IR systems described in section 1.2.3, although the difference is in the type of results retrieved for both IR systems. In this case, the system retrieves any type of structural unit instead of complete documents as in the case of traditional IR systems.

For example, if we introduce the query "income tax in Almería", the system could retrieve a section, a paragraph, a title, etc. which deal with this matter (corresponding all of them to structural units of the documents) instead of the whole documents.

1.3.3.2 Content and Structure queries

Content and Structure (CAS) queries may contain structural requirements. Such a query might arise if a user is aware of the document structure. To answer a CAS query a retrieval engine must deduce the information need from the query, identify elements that match structural requirements, and return the most relevant ones. For instance, we can take the query of the example of the previous section "income tax in Almería" adding a structural restricition that indicates the user is only

interested in sections commenting about this topic. Then, the system retrieves section units instead of any type of unit.

CAS queries may be represented by means of NEXI (Narrowed Extended XPath I), as we can see in [97]. NEXI is an IR query language for searching structured and semi-structured document collections. The language was first introduced for searching XML documents at INEX (see section 1.3.6.1) in 2004, and it has been used ever since.

The language is a tiny subset of XPath with an added `about()` function for identifying elements about some given topic, so we shall review the features of XPath relevant to our work.

Firstly, a name of an element, (A), selects all elements with that name (for example, *address*, in fig. 1.7, would select two nodes of the tree). A slash is used to select child nodes in the tree (A/B, where B is a direct descendant of A): *from/address* in the example. A double slash means that any number of elements could be included in the path ($A//B$, where B is a descendant, though not necessarily a direct one of A). For instance, *email//address* would select those units *address* which are directly or indirectly contained in an *email* element, i.e. with an arbitrary number of elements in between: *email/from/address* and *email/to/address*. A slash at the beginning of the expression means that the path starts at the root element ($/A$). An asterisk $*$ selects all the elements placed in the path after it ($/A//B//*$) (for example, */email//** would select all the descendants of *email*). Finally, a pair of opening and closing brackets and a number between them, after an element, establishes the order of the element as a child from left to right: $//A/B[3]$ means that the XPath expression engine would select the third B element child of A.

The language has extensions for question answering, multimedia searching, and searching heterogeneous document collections. Moreover, it is a language with a strict syntax.

1.3.4 Structured Information Retrieval models

Many of the structured Information Retrieval models, in concrete XML IR models, are an adaptation of the classical IR models shown in section 1.2.4, because

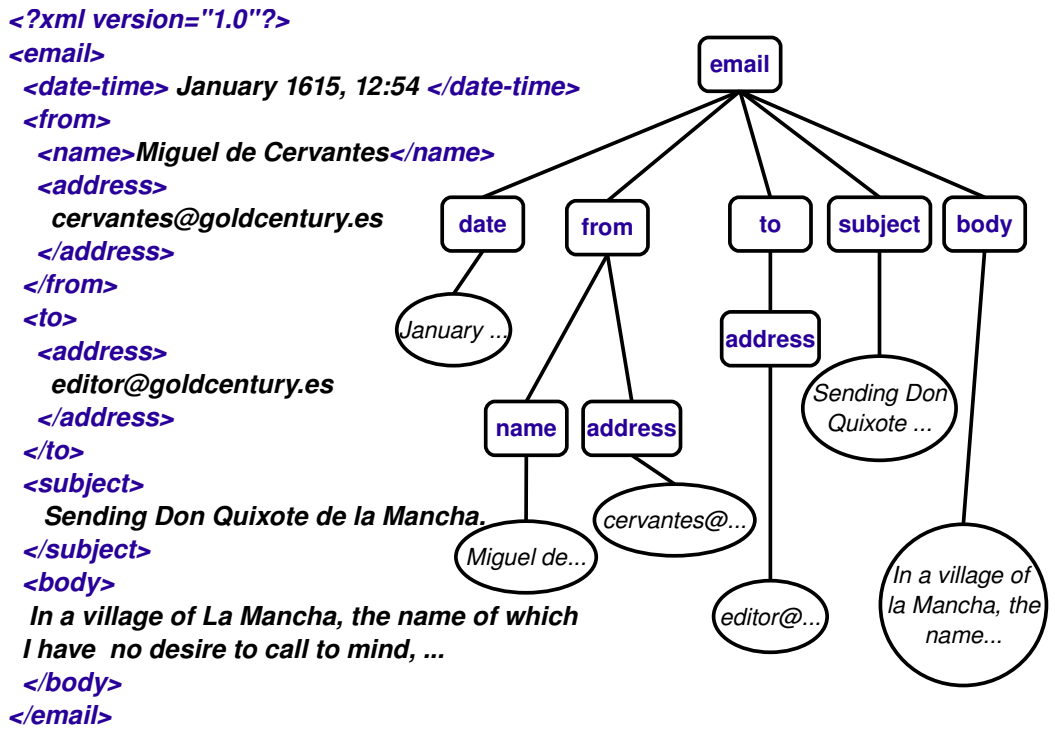


Figure 1.7: Example of XML document and its associated tree.

we can estimate the relevance of an element with these models using only its content. However, the use of other evidences like the context of the element is beneficial.

The used model depends on the indexing strategy to generate the ranking of the elements at all levels of granularity. Most of the following models are based on the estimation of the relevance of an element using content criterion. In addition, for a CAS query some additional processing is necessary to provide a ranking of elements which satisfies both content and structural criteria of the query. All these models are described in more detail in [60].

1.3.4.1 Element scoring

As we commented before, the scoring function is based on traditional IR models, such as the vector space model, which are adapted to XML retrieval. Then, it

gives an estimation of the element relevance for a given query. However, this score can be modified with additional strategies to generate the final ranking of elements.

1.3.4.2 Contextualization

Before introducing this approach, it is important to mention that many of the terms contained in some types of elements are a subset of those used in other types. Therefore, this term feature must be taken into account to estimate the relevance of elements down in the document hierarchy, improving this process.

This problem can be solved considering the context of the element (its parent, all or some of its ancestor, the root element) to estimate its relevance. The context of an element provides more evidence on what an element is or is not about. For instance, if an element does not contain all the query terms but the document where it is contained contains all the query terms, it is likely to be more relevant than this element contained in another document that does not contain all the query terms.

This methodology is succesful compared to the previous methodology (scoring the elements without context), specially for long documents.

1.3.4.3 Propagation

This model is used in combination with the leaf-only indexing strategy (see section 1.3.2). Then, the relevance of leaf elements for a query are estimated on the indexing process, resulting in retrieval scores for leaf elements. Afterwards, a propagation methodology is used to get the relevance of non-leaf elements based on the relevance of their descendant elements. This methodology follows the hierarchical structure from the leaf elements to the root element.

In general, the propagation model has a good retrieval performance.

1.3.4.4 Aggregation

The representation of each XML element in this model is based on the aggregation of its own content (if any) and the content of its children elements of the

hierarchical structure (if any). Retrieval is then based on aggregated representations. If we compare this model with propagation model, we can distinguish that the combination is applied to representations in this model, and it is applied to retrieval scores in the propagation model.

The representation of the element own content is generated using standard indexing techniques, whereas the representation of the non-leaf elements is generated by means of an aggregation function. There are different parameters which can be included in the aggregation function to indicate the influence of the children elements in the representation of the element.

1.3.4.5 Merging

This approach is combined with the distributed indexing strategy (see section 1.3.2), in which a separate index is created for each element type. Then, a retrieval model is used to generate the ranking of elements in each index, for instance the vector space model. Once all the rankings are generated, the lists are merged, but it is necessary a normalization to take into account the difference in size of the elements in the different indexes. According to this normalization, the scores across indexes are comparable, so the elements can be merged based on the normalizated scores.

Another approach where merging is used is when several rankings are generated for all elements in the collection, in contrast to separate indexes. The difference comes with the use of several strategies, each producing a ranking.

1.3.4.6 Processing structural constraints

In this section, we are going to describe approaches that were developed to process structural constraints expressed within NEXI.

A first approach consists of building a dictionary of tag synonyms. The dictionary can be syntactically based, for instance, `<p>` corresponds to a paragraph and `<p1>` corresponds to the first paragraph in a sequence, it would be logical to consider both tags are equivalent. It can be semantically based, for instance, `<house>` and `<home>` are equivalent tags. The dictionary can also be built from analyzing past relevance assessments, for instance, if a query asked for `<section>`

elements and the relevance assessments show that there are other types of elements which were retrieved for this query too, then they are considered equivalent to `<section>` element. Therefore, in the query asking for `<section>` elements, these types of elements could be retrieved too if the content criterion is satisfied.

Another technique is that of structure boosting. There, the structural constraints are not taken into account and the retrieval score of an element is based on the content. Then, it is boosted according to how the structural constraint of the query is satisfied by the element. The structure score is generated depending on the level of vagueness, for instance, the whole paths are compared or only the tags are compared. The structure boosting is also used when the propagation retrieval model (see section 1.3.4.3) is used to score non-leaf elements. In the propagation from leaf elements to parent elements, it is possible to boost the resulting score when the element matches the structural constraint. An important matter is to determine the level of vagueness.

1.3.5 Presenting results

After the structured IR system has retrieved the list of results sorted by their relevance degree, the next step is to show the ranking following different criteria. It is not common to show this list as it is retrieved from the search engine due to the XML elements are not independent like documents in traditiontal IR because they may overlap or may be siblings. These topics must be taken into account when the results are shown. Then, we can distinguish four different tasks to show the results in structured IR. All these tasks have been used in different INEX tracks (see section 1.3.6.1).

- *Thorough task*: It presents all the results from the search engine without any restriction. Therefore, we can find a lot of redundant information in the results as some of the elements are contained within other ones.

- *Focused task*: This task asks systems to return a ranked list of the most focused document parts (XML elements), where the resulting document parts should not overlap. For example, in the case of returning XML elements, a paragraph and its container section should not both be returned.

- In *context tasks*: These tasks correspond to end-user tasks where focused retrieval answers are grouped per document, in their original document order, providing access through further navigational means. This assumes that users consider documents as the most natural units of retrieval, and prefer an overview of relevance in their context. Two in context tasks are distinguished in the previous editions of INEX, depending on whether a set of document parts or a single answer part are returned per document.

  - *Relevant in context*: This task asks systems to return non-overlapping relevant document parts (XML elements or passages) clustered by the unit of the document that they are contained within.
  - *Best in context*: This task asks systems to return a single document part (XML element) per document. The single document part corresponds to the best entry point for starting to read the relevant text in the document.

1.3.6 Retrieval evaluation

As in the case of traditional IR, there exist different test collections for retrieval evaluation in structured IR field. This is the case of the following two XML collections: the **Shakespeare** collection[1] was initially used, however *INEX* (Initiative for the Evaluation of XML Retrieval) collection[2] is the standard one for structured IR.

1.3.6.1 Initiative for the Evaluation of XML retrieval

In the case of XML retrieval, it is necessary a test collection to evaluate the effectiveness of this type of systems where the relevance assesments are provided for different relevance criteria which take into account the structured organization of the collection.

Initiative for the Evaluation of XML retrieval (INEX) was set up in 2002 to address these issues. It consists of a forum for researchers to evaluate their

[1] available in `http://xml.coverpages.org/bosakShakespeare200.html`
[2] available in `http://www.inex.otago.ac.nz/`

content-oriented XML retrieval system and compare their results. The main idea of INEX is to establish an infrastructure and provide means (XML test collection and scoring methods) for the evaluation of these systems.

The main features of the XML test collection are: the document collection contains documents in XML format, there are two types of queries (Content-only, and Content and Structure queries) in the collection, and the relevance assesments are made on the XML element level. Moreover, relevance is measured to quantify if the system retrieves the right structural unit of the documents.

From 2002 to 2005, the document collection was composed of 12,107 articles of the *IEEE* Computer Society's publications from 12 magazines and 6 transactions, covering the period of 1995-2002, and totalling 494 megabytes in size. However, the collection was changed in 2006 using *Wikipedia* (and XML version of the English Wikipedia) with its 659,388 articles and around 4,600 megabytes in size, instead.

The main retrieval task to be performed in INEX is the *adhoc* retrieval of XML documents. In IR literature, *adhoc* retrieval is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. Until 2006, this task consisted of the *thorough task* retrieving a ranked list of document parts without any restriction, but this task was divided into other three tasks, namely *focused*, *relevant in context* and *best in context* in 2007, described in the previous section.

With regards to evaluation methodology, they adopted an evaluation framework where different aspects of focused retrieval can be consistently evaluated and compared. To measure the extent to which an XML retrieval system returns relevant information, they employed evaluation measures that only consider the amount of highlighted text in relevant documents. This is motivated by the need of directly exploiting the highlighting assessment procedure used at INEX, which leads to measures that are natural extensions of the well-established measures used in traditional IR.

1.3.6.2 Evaluation measures in INEX

The measures of retrieval effectiveness defined by Kamps et al. in [55] for the *adhoc* track of INEX 2007[1] and 2008[2], which are the editions objective of our experiments, are (these measures are contained in the *EvalJ*[3] project which develops java source code for the evaluation of information retrieval experiments):

- *Focused task*: Let p_r be the document part assigned to rank r in the ranked list of document parts L_q returned by a retrieval system for a topic q (at INEX, $|L_q| = 1500$ elements or passages). Let $rsize(p_r)$ be the length of highlighted (relevant) text contained by p_r in characters (if there is no highlighted text, $rsize(p_r) = 0$). Let $size(p_r)$ be the total number of characters contained by p_r, and let $Trel(q)$ be the total amount of (highlighted) relevant text for topic q. $Trel(q)$ is computed as the total number of highlighted characters across all documents, i.e., the sum of the lengths of the (non-overlapping) highlighted passages from all relevant documents. Precision at rank r is defined as the fraction of retrieved text that is relevant:

$$P[r] = \frac{\sum_{i=1}^{r} rsize(p_i)}{\sum_{i=1}^{r} size(p_i)}. \tag{1.1}$$

  To achieve a high precision score at rank r, the document parts retrieved up to and including that rank need to contain as little non-relevant text as possible. Recall at rank r is defined as the fraction of relevant text that is retrieved:

$$R[r] = \frac{\sum_{i=1}^{r} rsize(p_i)}{Trel(q)}. \tag{1.2}$$

  To achieve a high recall score at rank r, the document parts retrieved up to and including that rank need to contain as much relevant text as possible.

  An issue with the precision measure $P[r]$ given in eq. 1.1 is that it can be biased towards systems that return several shorter document parts rather

[1] http://inex.is.informatik.uni-duisburg.de/2007/
[2] http://www.inex.otago.ac.nz/
[3] http://evalj.sourceforge.net/README.html

than returning one longer part that contains them all. Since the notion of ranks is relatively fluid for passages, they opt to look at precision at recall levels rather than at ranks.

Specifically, they use an interpolated precision measure $iP[x]$, which computes interpolated precision scores at selected recall levels:

$$iP[x] = \begin{cases} max_{1 \leq r \leq |L_q|}(P[r] \wedge R[r] \geq x) & if\ x \leq R[|L_q|] \\ 0 & if\ x > R[|L_q|] \end{cases} \tag{1.3}$$

where $R[|L_q|]$ is the recall over all documents retrieved. For example, $iP[0.01]$ computes interpolated precision at the 1% recall level for a given topic.

Over a set of topics, they can also compute the interpolated precision measure, also denoted by $iP[x]$, by computing the mean of the scores obtained by the measure for each individual topic.

In addition to using the interpolated precision measure at selected recall levels, they also compute overall performance scores based on the measure of average interpolated precision AiP. For an INEX topic, they compute AiP by averaging the interpolated precision scores computed at the 101 standard recall levels (0.00, 0.01, . . . , 1.00):

$$AiP = \frac{1}{101} \cdot \sum_{x=0.00,0.01,\ldots,1.00} iP[x]. \tag{1.4}$$

Performance across a set of topics is measured by computing the mean of the AiP values obtained by the measure for each individual topic, resulting in mean average interpolate precision ($MAiP$). Assuming there are n topics:

$$MAiP = \frac{1}{n} \cdot \sum_{t} AiP[t]. \tag{1.5}$$

After defining the evaluation measures used in this task, the selected measures used in our experiments will be the interpolated precision (iP) at se-

lected recall levels (iP[0.0], iP[0.01], iP[0.05] and iP[0.10]) and the average interpolated precision (AiP) for the 101 recall levels.

- *Relevant in context and best in context*: The evaluation of the relevant in context and best in context task is based on the measures of generalized precision and recall, where the per document score reflects how well the retrieved text matches the relevant text in the document. The resulting measure was introduced at INEX 2006.

  To compute the score per document in the relevant in context task, for a retrieved document, the text identified by the selected set of non-overlapping retrieved parts is compared to the text highlighted by the assessor. More formally, let d be a retrieved document, and let p be a document part in d. They denote the set of all retrieved parts of document d as $\mathcal{P}_d$. Let $Trel(d)$ be the total amount of highlighted relevant text in the document d. $Trel(d)$ is computed as the total number of highlighted characters in a document, i.e., the sum of the lengths of the (non-overlapping) highlighted passages.

  They compute the following for a retrieved document d:

  - Document precision, as the fraction of retrieved text (in characters) that is (relevant):

    $$P(d) = \frac{\sum_{p \in \mathcal{P}_d} rsize(p)}{\sum_{p \in \mathcal{P}_d} size(p)}. \tag{1.6}$$

    The $P(d)$ measure ensures that, to achieve a high precision value for the document d, the set of retrieved parts for that document needs to contain as little non-relevant text as possible.

  - Document recall, as the fraction of highlighted text (in characters) that is retrieved:

    $$R(d) = \frac{\sum_{p \in \mathcal{P}_d} rsize(p)}{Trel(d)}. \tag{1.7}$$

The $R(d)$ measure ensures that, to achieve a high recall value for the document d, the set of retrieved parts for that document needs to contain as much relevant text as possible.

– Document F-Score, as the combination of the document precision and recall scores using their harmonic mean [98], resulting in a score in [0,1] per document:

$$F(d) = \frac{2 \cdot P(d) \cdot R(d)}{P(d) + R(d)}. \tag{1.8}$$

For retrieved non-relevant documents, both document precision and document recall evaluate to zero.

They may choose either precision, recall, the F-score, or even other aggregates as document score (S(d)). For the relevant in context task, they use the F-score as the document score:

$$S(d) = F(d). \tag{1.9}$$

The resulting S(d) score varies between 0 (document without relevant text, or none of the relevant text is retrieved) and 1 (all relevant text is retrieved without retrieving any non-relevant text).

However, for the best in context task the document score S(d) is computed with a distance similarity measure, $s(x, b)$, which measures how close the system-proposed entry point x is to the ground-truth best entry point b given by the assessor. Closeness is assumed to be an inverse function of distance between the two points. The maximum value of 1 is achieved when the two points match, and the minimum value is zero.

They use the following formula for computing the distance similarity measure:

$$s(x, b) = \begin{cases} \frac{n - d(x,b)}{n} & if\ 0 \leq d(x, b) \leq n \\ 0 & if\ d(x, b) > n \end{cases} \tag{1.10}$$

where the distance $d(x, b)$ is measured in characters, and n is the number of characters representing the visible part of the document that can fit on a screen (typically, $n = 1000$ characters).

They use the $s(x, b)$ distance similarity score as the document score for the best in context task:

$$S(d) = s(x, b). \tag{1.11}$$

The resulting $S(d)$ score varies between 0 (non-relevant document, or the distance between the system-proposed entry point and the ground-truth best entry point is more than n characters) and 1 (the system-proposed entry point is identical to the ground-truth best entry point).

After defining the score of the document for both tasks, the system computes the scores for ranked list of documents.

Given that the individual document scores ($S(d)$) for each document in a ranked list $\mathcal{L}$ can take any value in [0,1], they employ the evaluation measures of generalized precision and recall.

More formally, let us assume that for a given topic there are in total $Nrel$ relevant documents, and let $IsRel(dr) = 1$ if document d at document-rank r contains highlighted relevant text, and $IsRel(dr) = 0$ otherwise. Let $Nrel$ be the total number of document with relevance for a given topics.

Over the ranked list of documents, they compute the following:

- generalized precision ($gP[r]$), as the sum of document scores up to (and including) document-rank r, divided by the rank r:

$$gP[r] = \frac{\sum_{i=1}^{r} S(d_i)}{r}. \tag{1.12}$$

- generalized Recall ($gR[r]$), as the number of relevant documents retrieved up to (and including) document-rank r, divided by the total

number of relevant documents:

$$gR[r] = \frac{\sum_{i=1}^{r} IsRel(d_i)}{Nrel}. \tag{1.13}$$

Based on these, the average generalized precision AgP for a topic can be computed by averaging the generalized precision scores obtained for each natural recall points, where generalized recall increases:

$$AgP = \frac{\sum_{r=1}^{|\mathcal{L}|} IsRel(d_r) \cdot gP[r]}{Nrel}. \tag{1.14}$$

For non-retrieved relevant documents a generalized precision score of zero is assumed.

The mean average generalized precision ($MAgP$) is simply the mean of the average generalized precision scores over all topic.

After defining the evaluation measures used in this task, the selected measures used in our experiments will be the generalized precision in different ranges (5,10,15 and 20) and the average generalized precision (AgP).

Chapter 2

The context-based influence diagram approach for XML retrieval

2.1 Introduction

Classical probabilistic IR systems rank the documents by considering their probability of relevance to a given query. In these systems, the action of retrieving (or not) a document is independent on the action of retrieving (or not) any other document. However, this is no longer true when dealing with structured documents, where the decision about retrieving a document component clearly may affect the retrieval of other components.

For example, it makes no sense to retrieve two sections of a chapter and also the complete chapter itself. Therefore, it is clear that not only the probability of relevance has to be used to retrieve the document components, but also the *usefulness* of these components for the user, taking into account the context where they are placed and what has been previously retrieved.

Following this direction, the *Context-based Influence Diagram* model for structured documents (CID model), described by Campos et al. in [23], was born with the capability of making decisions about which document components should be retrieved, not only depending on their probabilities of relevance, but also on their

utilities for the user and the influences provided by the context in which each structural component is located.

Then, these parameters correspond to two types of information combined to get the relevance degree of each component in a document. On the one hand, the specificity of the component with respect to the query: the more terms in the component appear in the query, the more relevant becomes the component, that is to say, the more clearly the component is only about (at least a part of) the topic of the query. On the other hand, the exhaustivity of the component with respect to the query: the more terms in the query match with the terms in the component, the more relevant the component is, i.e., the more clearly the component comprises the topic of the query. The components that best satisfy the user's information need expressed by means of the query should be, simultaneously, as specific and exhaustive as possible.

This is carried out by means of an *Influence Diagram* (ID) [53], a generalization of the well founded *Bayesian Network* (BN) formalism [73] in the context of Decision Theory [34]. Examples in the specialized literature about the application of BNs to structured IR are [18, 45, 71, 76], although the CID model is the only one, as far as we know, that applies IDs.

Thus, the CID model is based on the IDs formalism providing a visual representation of a decision problem. BNs offer an intuitive way to identify and display the essential elements of the domain (the structured document components and their usefulness) and also how they are related to each other. They have also associated quantitative knowledge that measures the strength of the interactions. Moreover, this model takes into account the influences provided by the context in which each structural component is located. By means of this approach, structured retrieval task is presented as a decision-making problem.

Solving an ID means to determine the expected utility of each one of the possible decisions, for those situations of interest, with the aim of making decisions which maximize the expected utility, as Shachter describes in [88]. The expected utility in the CID model depends on the bi-dimensional posterior probabilities, corresponding to each structural unit and the unit where it is contained. In [23], and in order to simplify the computations, it was assumed that the two involved units were independent given the query, so the bi-dimensional distributions could

be approximated just multiplying the unidimensional posterior probabilities of each unit given the query. In this chapter, this efficient approximated evaluation method is presented instead of the general method.

Lastly, the *Garnata* IR system has been specifically designed to implement models based on PGMs like the CID model in our case. In its development, the underlying model to deal with structured documents was carefully studied, extracting the most critical operations in terms of efficiency. Having them in mind, the system has been implemented with the aim of optimizing them, using an efficient combination of data structures that assure a good response time for a query.

In order to describe precisely these ideas and formalize them, this chapter is organized in the following way: In section 2.2 we briefly introduce some preliminary concepts: we provide some background about BNs and IDs. Section 2.3 describes the type of structured documents being considered. The following two sections introduce the model: Section 2.4 presents the BN that represents graphically the structure of the documents, and the corresponding ID is described in section 2.5. Section 2.6 explains how to use the model for retrieval purposes by computing the expected utilities of the document components. Lastly, section 2.8 describes the IR system (Garnata) which implements the CID model.

2.2 Preliminaries

2.2.1 Bayesian networks

In formal terms, a BN is a directed acyclic graph (a graph with links which are orientated, taking the name of arcs, and with no cycles in it), in which the nodes represent random variables and the arcs show causality or dependency relationships between them.[1] The variables and their relationships comprise the qualitative knowledge stored in a BN, we can see an example in fig. 2.1. A second type of knowledge also stored in the BN is known as quantitative, since it establishes the strength of the relationships and is measured by means of probability

[1] A dependence relationship between two variables, X and Y , implies a modification of the belief in X, given that the value taken by Y is known. An Independence relationship means that the belief in X is not modified, given the knowledge on Y.

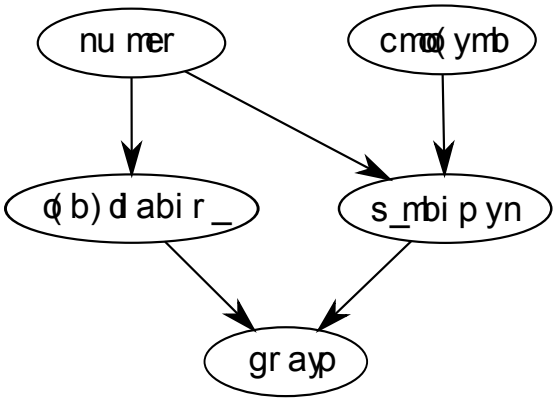


Figure 2.1: An example of a Bayesian Network.

distributions. Associated with each node there is a set of conditional probability distributions, one for each possible combination of values that its parents can take.

Formally, a BN can be considered an efficient representation of a joint probability distribution that takes into account the set of independence relationships represented in the graphical component of the model. In general terms, given a set of variables $\{X_1, ..., X_n\}$ and a BN G, the joint probability distribution in terms of local conditional probabilities is obtained as follows

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)),$$

where $Pa(X_i)$ is the parent set of X_i in the graph. If X_i has no parents, then the set $Pa(X_i)$ is empty, and therefore $P(X_i|Pa(X_i))$ is just $P(X_i)$.

Once completed, a BN can be used to derive the posterior probability distribution of one or more variables since we have observed the particular values for other variables in the network, or to update previous conclusions when new evidence reach the system. Researchers have developed general inference algorithms that take advantage of the independences represented in the network. Although it is possible to find algorithms that perform inference tasks in a time that is linear in the number of variables, high computational complexity inference algorithms result from having multiple pathways connecting nodes in the graph. General inference has been proved to be NP-hard, as we can see in [17].

2.2.2 Influence diagrams

IDs [53, 89] are probabilistic graphical models that provide a simple notation for designing decision models by clarifying the qualitative issues of what factors need to be considered and how they are related, i.e. an intuitive representation of the model. They have also associated an underlying quantitative representation in order to measure the strength of the relationships: we can quantify uncertain interactions among random variables and also the decision maker's options and preferences. The model is used to determine the optimal decision policy. IDs contain three types of nodes:

- *Decision nodes* (drawn as rectangles) represent variables that the decision maker controls directly, and model the decision alternatives available for the decision maker.
- *Chance nodes* (drawn as circles) represent random variables, i.e., uncertain quantities that are relevant to the decision problem and can not be controlled directly. They are quantified by means of conditional probability distributions.
- *Utility nodes* (drawn as diamonds) represent utility, i.e., express the profit or the preference degree of the consequences derived of the decision process. They are quantified by the utility of each of the possible combinations of outcomes of their parent nodes.

There are also different types of arcs in an ID, which generally represent influence. The arcs between chance nodes represent probabilistic dependences (as occurs in BNs). The arcs from a decision node to a chance node or to a utility node establish that the future decision will affect the value of the chance node or the profit obtained, respectively. Arcs between a chance node and a decision node (also called *informative*) only say that the value of the chance node will be known at the moment of making the decision. Finally, arcs from a chance node to a utility node will represent the fact that the profit depends on the value that this chance node takes. The absence of an arc between two nodes specifies (conditional) independence relationships. It should be noted that the absence of

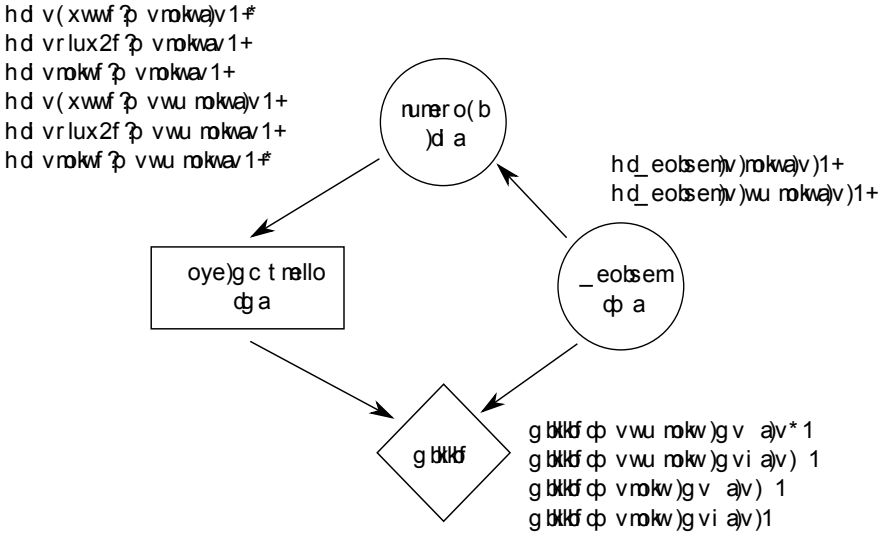


Figure 2.2: An example of an Influence Diagram.

an arc is a stronger statement than the presence of an arc, which only indicates the possibility of dependence.

Some arcs in IDs clearly have a causal meaning. In particular, a directed path from a decision node to a chance node means that the decision will influence that chance node, in the sense of changing its probability distribution.

A simple example of an ID appears in fig. 2.2. It has two chance nodes, F and W, representing, the weather forecast in the morning (sunny, cloudy or rainy), and whether it actually rains during the day (rain or no-rain), respectively. It has one decision node U, take an umbrella (with possible values true or false). The utility node measures the decision maker's satisfaction.

With each chance node X in the graph, the quantitative part of an ID associates a set of conditional probability distributions $P(X|pa(X)))$, one for each configuration $pa(X)$ from the parent set of X in the graph, $Pa(X)$, following the BN philosophy. For each utility node V, a set of utility values $v(pa(V))$ is associated, specifying for each combination of values for the parents of V, a number expressing the desirability of this combination for the decision maker.

The goal of ID modeling is to choose the decision alternative that will lead to the highest expected gain (utility), i.e. to find the optimal policy [88, 108].

In order to compute the solution, for each sequence of decisions, the utilities of its uncertain consequences are weighted with the probabilities that these consequences will occur.

2.3 Type of structured documents

A document collection is composed of M *documents*, $\mathcal{D} = \{D_1, \ldots, D_M\}$, and the set of the *terms* used to index these documents (the glossary of the collection). Moreover, each document D_i is organized hierarchically, representing structural associations of elements in D_i, which will be called *structural units*. Each structural unit is composed of other smaller structural units, except some 'terminal' or 'minimal' units which are indivisible, they do not contain any other unit. Instead, these are composed of terms: each term used to index the complete document D_i will be assigned to all the terminal units containing it. Conversely, each structural unit, except the one corresponding to the complete document, is included in only one structural unit. Therefore, the structural units associated to a document D_i form a (inverted) tree.

For instance, a scientific article may contain a title, authors, abstract, sections and bibliography; sections may contain a title, subsections and paragraphs; in turn subsections contain paragraphs and perhaps also a title; the bibliography contain references; titles, authors, paragraphs, abstract and references would be in this case the terminal structural units (see fig. 2.3).

2.4 The underlying Bayesian Network

The BN will contain two types of nodes, representing the terms and the structural units. The former will be represented by the set $\mathcal{T} = \{T_1, T_2, \ldots, T_l\}$. There are two types of structural units: *basic structural units*, those which only contain terms, and *complex structural units*, that are composed of other basic or complex units. The notation for these nodes is $\mathcal{U}_b = \{B_1, B_2, \ldots, B_m\}$ and $\mathcal{U}_c = \{S_1, S_2, \ldots, S_n\}$, respectively. Therefore, the set of all structural units is $\mathcal{U} = \mathcal{U}_b \cup \mathcal{U}_c$. In this chapter, T or T_k will represent a term; B or B_i a basic structural unit, and S or S_j a complex structural unit. Generic structural

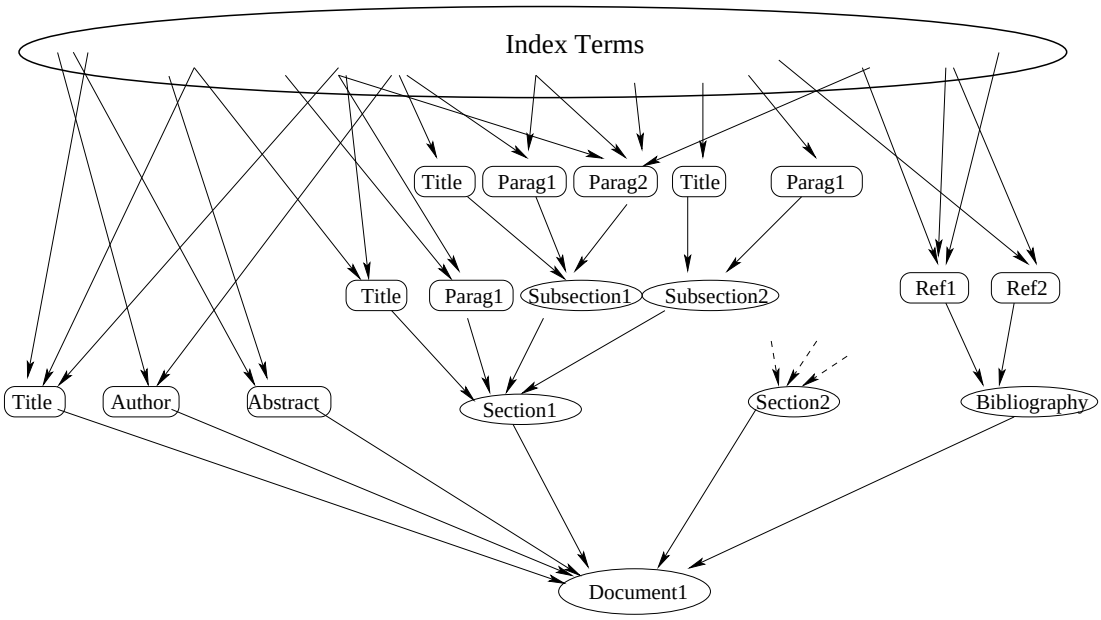


Figure 2.3: Example of the structure of a scientific article.

units (either basic or complex) will be denoted as U_i or U. Each node T, B or S has associated a binary random variable, which can take its values from the sets $\{t^-, t^+\}$, $\{b^-, b^+\}$ or $\{s^-, s^+\}$ (the term/unit is not relevant or is relevant), respectively. A unit is relevant for a given query if it satisfies the user's information need expressed by this query. A term is relevant in the sense that the user believes that it will appear in relevant units/documents.

Regarding the arcs of the model, there is an arc from a given node (either term or structural unit) to the particular structural unit node it belongs to, expressing the fact that the relevance of a given structural unit to the user will depend on the relevance values of the different elements (units or terms) that comprise it. It should be noted that with this criteria, terms nodes have no parents. Formally, the network is characterized by the following parent sets, $Pa(.)$:

- $\forall T \in \mathcal{T}$, $Pa(T) = \emptyset$.
- $\forall B \in \mathcal{U}_b$, $\emptyset \neq Pa(B) \subseteq \mathcal{T}$.
- $\forall S \in \mathcal{U}_c$, $\emptyset \neq Pa(S) \subseteq \mathcal{U}_b \cup \mathcal{U}_c$, with $Pa(S_1) \cap Pa(S_2) = \emptyset$, $\forall S_1 \neq S_2 \in \mathcal{U}_c$.

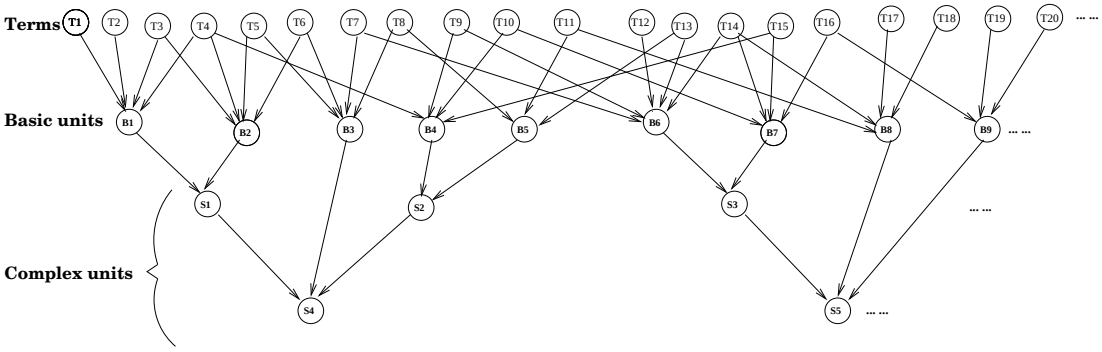


Figure 2.4: Bayesian Network representing a structured document collection.

It should be noticed that the hierarchical structure of the model determines that each structural unit $U \in \mathcal{U}$ has only one structural unit as its child, the unique structural unit containing U (except for the leaf nodes, i.e. the complete documents, which have no child). We shall denote indistinctly by $Hi(U)$ or $U_{hi(U)}$ the single child node associated with node U (with $Hi(U) = null$ if U is a leaf node). Fig. 2.4 displays an example of the network topology.

The numerical values for the conditional probabilities have also to be assessed: $p(t^+)$, $p(b^+|pa(B))$, $p(s^+|pa(S))$, for every node in $\mathcal{T}$, $\mathcal{U}_b$ and $\mathcal{U}_c$, respectively, and every configuration of the corresponding parent sets ($pa(X)$ denotes a configuration or instantiation of the parent set of X, $Pa(X)$). Once specified, the network may be used to compute the posterior probabilities of relevance of all the structural units $U \in \mathcal{U}$ for a given query.

In our case, the number of terms and structural units considered may be quite large (thousands or even hundred thousands). Moreover, the topology of the BN contains multiple pathways connecting nodes (because the terms may be associated to different basic structural units) and possibly nodes with a great number of parents (so that it can be quite difficult to assess and store the required conditional probability tables). For these reasons the canonical model is used to represent the conditional probabilities proposed in [22] (as the CID model does), which supports a very efficient inference procedure. Considering the conditional probabilities for the basic structural units, having a subset of terms as their parents, and for the complex structural units, having other structural units as

their parents. These probabilities are defined as follows:

$$\forall B \in \mathcal{U}_b,\ p(b^+|pa(B)) = \sum_{T \in R(pa(B))} w(T,B)\,, \tag{2.1}$$

$$\forall S \in \mathcal{U}_c,\ p(s^+|pa(S)) = \sum_{U \in R(pa(S))} w(U,S)\,, \tag{2.2}$$

where $w(T,B)$ is a weight associated to each term T belonging to the basic unit B, $w(U,S)$ is a weight measuring the importance of the unit U within S. In any case $R(pa(U))$ is the subset of parents of U (terms for B, and either basic or complex units for S) relevant in the configuration $pa(U)$, i.e., $R(pa(B)) = \{T \in Pa(B)\,|\,t^+ \in pa(B)\}$ and $R(pa(S)) = \{U \in Pa(S)\,|\,u^+ \in pa(S)\}$. So, the more parents of U are relevant the greater the probability of relevance of U. For example, for the unit B_2 in the model displayed in fig. 2.4, $Pa(B_2) = \{T_3, T_4, T_5, T_6\}$; if the configuration $pa(B_2) = \{t_3^+, t_4^-, t_5^+, t_6^-\}$, then $p(b_2^+|pa(B_2)) = w(T_3,B_2) + w(T_5,B_2)$.

The weights can be defined in any way, the only restrictions are that $w(T,B) \geq 0$, $w(U,S) \geq 0$, $\sum_{T \in Pa(B)} w(T,B) \leq 1$, and $\sum_{U \in Pa(S)} w(U,S) \leq 1$. For example, they can be defined using a normalized tf–idf scheme, as in [23], or the relative importance of each type of unit could also be considered (for example, the title or the abstract could be more representative of the content of a document than a section).

For the tf–idf scheme, before defining the weights $w(T,B)$ and $w(U,S)$ in eqs. 2.1 and 2.2, let us introduce some additional notation: for any unit $U_i \in \mathcal{U}$, let $A(U_i) = \{T_k \in \mathcal{T} | T_k \textit{ is an ancestor of } U_i\}$, i.e., $A(U_i)$ is the set of terms that are included in the unit U_i.[1] For example, for the model displayed in fig. 2.4, $A(S_2) = \{T_4, T_8, T_9, T_{10}, T_{11}, T_{13}, T_{15}\}$ and $A(B_3) = \{T_5, T_6, T_7, T_8\}$. Let $\rho(T_k, C)$ be the weighting scheme where $\rho(T_k, C) = tf_{k,C} \cdot idf_k$ being $tf_{k,C}$ the *frequency* of the term T_k (number of times that T_k occurs) in the set of terms C and idf_k the *inverse document frequency* of T_k in the whole collection. Then the weight is

[1] Two facts should be noted: first, that although a unit S_i is not connected directly to any term, it does contain all the terms indexing structural basic units that are included in S_i; and secondly, $A(B_i) = Pa(B_i)$.

defined as follows:

$$\forall B_i \in \mathcal{U}_b, \forall T_k \in Pa(B_i), w(T_k, B_i) = \frac{\rho(T_k, A(B_i))}{\sum_{T_h \in A(B_i)} \rho(T_h, A(B_i))}, \tag{2.3}$$

$$\forall S_i \in \mathcal{U}_c, \forall U_h \in Pa(S_i), w(U_h, S_i) = \frac{\sum_{T_k \in A(U_h)} \rho(T_k, A(U_h))}{\sum_{T_k \in A(S_i)} \rho(T_k, A(S_i))}. \tag{2.4}$$

It should be observed that the weights in eq. 2.3 are only the classical tf–idf weights, normalized to sum one up. The weights $w(U_h, S_i)$ in eq. 2.4 measure, to a certain extent, the proportion of the content of the unit S_i which can be attributed to each one of its components.

With respect to the prior probabilities of relevance of the terms, $p(t^+)$, they can also be defined in any reasonable way, for example an identical probability for all the terms, $p(t^+) = p_0$, $\forall T \in \mathcal{T}$, as proposed in [23], where $p_0 = \frac{1}{|\mathcal{T}|}$ being $|\mathcal{T}|$ the number of terms in the collection of documents.

2.4.1 Virtual nodes

At the beginning of this section, we mentioned that each complex structural unit is composed of other smaller structural units, except for the basic units, which do not contain any other unit but text. However, it is quite common to find collections where a unit can contain text and other units at the same time; e.g. a paragraph which includes a bold-faced sentence:

```
<paragraph> This is an example of a paragraph with a <bold>
bold-faced text <\bold>. <\paragraph>
```

To deal with this situation, and in order to continue having a clear distinction between complex and basic units, so the propagation algorithm could work correctly, the model includes some special nodes, called *virtual units*. They are fictitious nodes that are parents of the unit where this situation has been detected (in the previous example, the 'paragraph' unit) together with the units contained in it (in this case, only 'bold'). Therefore, these virtual units will be basic units containing only terms. Fig. 2.5 shows this situation.

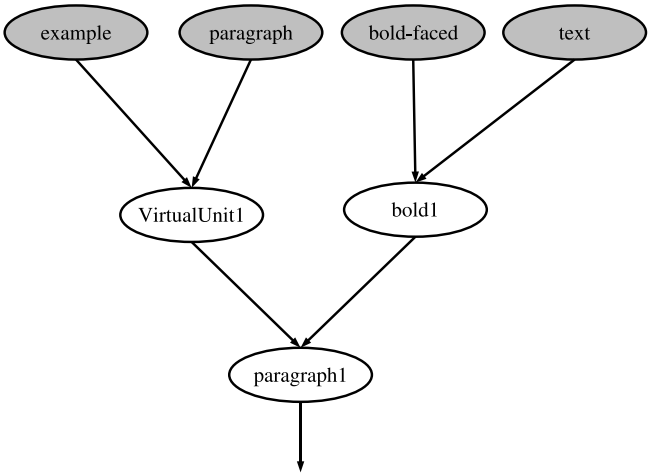


Figure 2.5: Fragment of Bayesian Network containing a virtual unit.

These virtual units cannot be retrieved and their corresponding weights are computed according to the general weighting scheme used, taking into account their parent terms.

2.5 The Influence Diagram model

Once the BN has been constructed, it is enlarged by including decision and utility nodes, thus transforming it into an ID. The same topology proposed by Campos et al. in [23] is used:

- *Decision nodes*: One decision node, R_i, for each structural unit $U_i \in \mathcal{U}$. R_i represents the decision variable related to whether or not to return the structural unit U_i to the user. The two different values for R_i are r_i^+ and r_i^-, meaning 'retrieve U_i' and 'do not retrieve U_i', respectively.

- *Utility nodes*: One of these, V_i, for each structural unit $U_i \in \mathcal{U}$, will measure the value of utility of the corresponding decision.

In addition to the arcs between chance nodes (already present in the BN), a set of arcs pointing to utility nodes are also included, employed to indicate which

variables have a direct influence on the desirability of a given decision, i.e. the profit obtained will depend on the value of these variables. In order to represent that the utility function of V_i obviously depends on the decision made and the relevance value of the structural unit considered, we use arcs from each chance node U_i and decision node R_i to the utility node V_i.

Another important set of arcs are those going from $Hi(U_i)$ to V_i, which represent that the utility of the decision about retrieving the unit U_i also depends on the relevance of the unit which contains it (obviously, for the units which are not contained in any other unit these arcs do not exist). This last type of arc allows us to represent the context-based information and can avoid redundant information being shown to the user. For instance, we could express the fact that on the one hand, if U_i is relevant and $Hi(U_i)$ is not, then the utility of retrieving U_i should be large (and the one of not retrieving it almost null). On the other hand, if $Hi(U_i)$ is relevant, even if U_i were also relevant the utility of retrieving U_i should be small because, in this case, it would be preferable to retrieve the largest unit as a whole, instead of each of its components separately.

Another utility node, denoted by Σ, that represents the joint utility of the whole model is also considered. It has all the utility nodes V_j as its parents. These arcs represent the fact that the joint utility of the model will depend (additively) on the values of the individual utilities of each structural unit. Fig. 2.6 displays an example of the topology of the ID being considered.

Moreover, the ID requires numerical values for the utilities which are necessary to compute the expected utility of retrieving structural units, $EU(r_i|q)$, namely $v(r_i, u_i, u_{hi(U_i)})$. For each utility node V_i we need eight numbers, one for each combination of values of the decision node R_i and the chance nodes U_i and $Hi(U_i)$ (except for the leaf nodes, which only require four values). These values are represented by $v(r_i, u_i, u_{hi(U_i)})$, with $r_i \in \{r_i^-, r_i^+\}$, $u_i \in \{u_i^-, u_i^+\}$, and $u_{hi(U_i)} \in \{u^-_{hi(U_i)}, u^+_{hi(U_i)}\}$.

The utilities are composed of a component which depends on the involved unit and another component independent on the specific unit and depending only on which one of the four configurations, $(u_i^-, u^-_{hi(U_i)})$, $(u_i^-, u^+_{hi(U_i)})$, $(u_i^+, u^-_{hi(U_i)})$ or

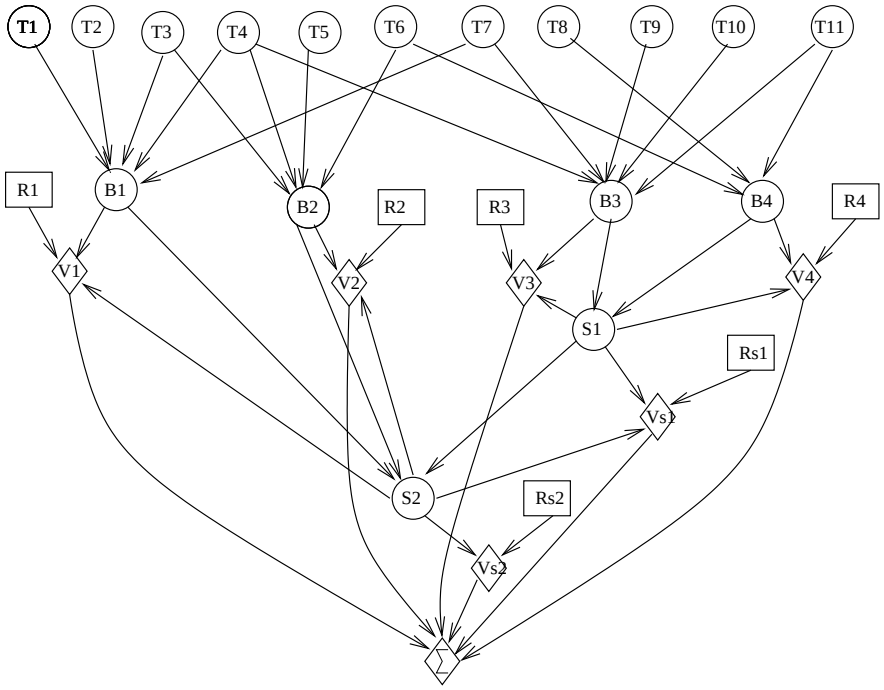


Figure 2.6: Topology of the Influence Diagram.

$(u_i^+ , u_{hi(Ui)}^+)$, is being considered:

$$v(r_i, u_i, u_{hi(U_i)}) = nidf_Q(U_i) \cdot v(u_i, u_{hi(U_i)}), \tag{2.5}$$

with $v(u_i^-, u_{hi(U_i)}^-) = v^{--}$, $v(u_i^-, u_{hi(U_i)}^+) = v^{-+}$, $v(u_i^+, u_{hi(U_i)}^-) = v^{+-}$ and $v(u_i^+, u_{hi(U_i)}^+) = v^{++}$.

The part depending on the involved unit is defined as the sum of the inverted document frequencies of those terms contained in U_i that also belong to the query Q, normalized by the sum of the idfs of the terms contained in the query (a unit U_i will be more useful or exhaustive, with respect to a query Q, as more terms indexing U_i also belong to Q):

$$nidf_Q(U_i) = \frac{\sum_{T \in An(U_i) \cap Q} idf(T)}{\sum_{T \in Q} idf(T)}. \tag{2.6}$$

2.6 Solving the Influence Diagram

To solve an ID, the expected utility of each possible decision (for those situations of interest) has to be computed, thus making decisions which maximize the expected utility. In our case, the situation of interest corresponds with the information provided by the user when he/she formulates a query. Let $\mathcal{Q} \subseteq \mathcal{T}$ be the set of terms used to express the query. Each term $T_i \in \mathcal{Q}$ will be instantiated to either t_i^+ or t_i^-; let q be the corresponding configuration of the variables in $\mathcal{Q}$. Then, the expected utility of each decision given q is computed. As a global additive utility model is defined, and the different decision variables R_i are not directly linked to each other, the model processes each independently. The expected utilities for each U_i are computed by means of

$$EU(r_i^+ \mid q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}}} v(r_i^+, u_i, u_{hi(U_i)})\, p(u_i, u_{hi(U_i)}|q)\,. \quad (2.7)$$

$$EU(r_i^- \mid q) = \sum_{\substack{u_i \in \{u_i^-, u_i^+\} \\ u_{hi(U_i)} \in \{u_{hi(U_i)}^-, u_{hi(U_i)}^+\}}} v(r_i^-, u_i, u_{hi(U_i)})\, p(u_i, u_{hi(U_i)}|q)\,. \quad (2.8)$$

In the context of a typical decision making problem, once the expected utilities are computed, the decision with greatest utility is chosen: this would mean to retrieve the structural unit U_i if $EU(r_i^+|q) \geq EU(r_i^-|q)$, and not to retrieve it otherwise. However, the purpose is not only to make decisions about what to retrieve, but also to give a ranking of those units. The simplest way to do it is to show them in decreasing order of the utility of retrieving U_i, $EU(r_i^+|q)$[1]. In this case only four utility values have to be assessed, and only eq. 2.7 is required.

2.7 Computing probabilities

In order to provide to the user an ordered list of structural units, the model computes the posterior probabilities of relevance of all the structural units $U \in \mathcal{U}$,

[1]Other options would also be possible to generate a ranking, as for example to use the difference between both expected utilities, $EU(r_i^+|q) - EU(r_i^-|q)$.

$p(u^+|q)$, and also the bi-dimensional posterior probabilities, $p(u^+, u^+_{hi(U)}|q)$[1]. Due to the specific characteristics of the canonical model used to define the conditional probabilities, the posterior probabilities are computed efficiently. These probabilities represent the specificity component of each structural unit U: the more terms indexing U also belong to q, the more probable is U.

2.7.1 Calculus of unidimensional posterior probabilities

Proposition 1.

$$\forall B \in \mathcal{U}_b,\ \ p(b^+|q) \ = \sum_{T \in Pa(B) \setminus Q} w(T,B)\, p(t^+) + \sum_{T \in Pa(B) \cap R(q)} w(T,B)\,. \quad (2.9)$$

$$\forall S \in \mathcal{U}_c,\ \ p(s^+|q) \ = \sum_{U \in Pa(S)} w(U,S)\, p(u^+|q)\,. \quad (2.10)$$

As we can see, the posterior probabilities of the basic units can be computed directly. The posterior probabilities of the complex units can be calculated in a top-down manner, starting from those for the basic units. However, it is possible to design a more direct inference method. We need some additional notation: $\forall S \in \mathcal{U}_c$, let $A_b(S) = \{B \in \mathcal{U}_b \,|\, B \text{ is an ancestor of } S\}$, $A_c(S) = \{S' \in \mathcal{U}_c \,|\, S' \text{ is an ancestor of } S\}$, and $\forall B \in \mathcal{U}_b$, let $D_c(B) = \{S \in \mathcal{U}_c \,|\, S \text{ is a descendant of } B\}$. Notice that, for each basic unit B in $A_b(S)$, there is only one path going from B to S. The weight $w(B,S)$ is defined as the product of the weights of the arcs in the path from B to S, i.e. $w(B,S) = w(B, Hi(B)) \prod_{S' \in A_c(S) \cap D_c(B)} w(S', Hi(S'))$. Then, the result is:

Proposition 2.

$$\forall S \in \mathcal{U}_c,\ p(s^+|q) = \sum_{B \in A_b(S)} w(B,S)\, p(b^+|q)\,. \quad (2.11)$$

Proposition 2 states that the posterior probability of a complex structural unit S is computed by calculating the average of the posterior probabilities of all the basic structural units B contained in S, each probability being weighted by the

[1]Notice that the other required bi-dimensional probabilities, $p(u^+, u^-_{hi(U)}|q)$, $p(u^-, u^+_{hi(U)}|q)$ and $p(u^-, u^-_{hi(U)}|q)$, can be easily computed from $p(u^+, u^+_{hi(U)}|q)$, $p(u^+|q)$ and $p(u^+_{hi(U)}|q)$.

product of the weights of the arcs in the (single) path going from B to S. This result is the basis to develop an inference process able to compute all the posterior probabilities of the structural units in a single traversal of the graph, starting only from the instantiated terms in $\mathcal{Q}$, provided that the prior probabilities of relevance have been calculated and stored within the structure:

Proposition 3.

$$\forall B \in \mathcal{U}_b,\ p(b^+|q) = p(b^+) + \sum_{T \in Pa(B) \cap R(q)} w(T,B)\left(1 - p(t^+)\right) - \sum_{T \in Pa(B) \cap (Q \setminus R(q))} w(T,B)\, p(t^+)\,. \quad (2.12)$$

$$\forall S \in \mathcal{U}_c,\ p(s^+|q) = p(s^+) + \sum_{\substack{B \in A_b(S) \\ Pa(B) \cap Q \neq \emptyset}} w(B,S)\ \left(p(b^+|q) - p(b^+)\right)\,. \quad (2.13)$$

This result indicates how the posterior probabilities are computed from the prior probabilities traversing the nodes in the graph that will require updating. An algorithm that computes all the posterior probabilities $p(b^+|q)$ and $p(s^+|q)$, based on Proposition 3, starts from the terms in $\mathcal{Q}$ and carries out a width graph traversal until it reaches the basic units that require updating, computing $p(b^+|q)$ using eq. 2.12. Starting from these modified basic units, it carries out a depth graph traversal to compute $p(s^+|q)$, only for those complex units that require updating, using eq. 2.13. This algorithm needs the previous computation and storage of the nodes' prior probabilities. This can be done easily using Propositions 1 and 2 (with $q = \emptyset$).

2.7.2 Approximating the bi-dimensional posterior probabilities

As Campos et al. describe in [24], the CID model permits computing exactly the bi-dimensional probabilities involved in the computation of the expected utilities. But this process could be expensive in terms of memory and time for very

large document collections. Therefore, an approximation was proposed in [23] which assumes the independence between each structural unit and the one which contains it given the query.

$$p(u^+, u^+_{hi(U)}|q) = p(u^+|q)p(u^+_{hi(U)}|q). \tag{2.14}$$

Then, the computation of the bi-dimensional posterior probabilities only requires the values of the unidimensional probabilities explained in the previous section.

2.8 The Garnata information retrieval system for XML documents

Garnata was born as an implementation completely adapted to the models based on the above probabilistic graphical models to retrieve structured documents, although other models following the same philosophy could be easily implemented in it. Afterwards, we can emphasize that the CID model, which has been the objective of this chapter, has been implemented in Garnata.

Written in C++, following the object-oriented paradigm, it offers a wide range of classes and a complete set of utility programs.

In the following sections we shall study its architecture from the indexing and querying points of view, presenting its main features.

2.8.1 Indexing subsystem

2.8.1.1 Application level

Garnata is able to manage different collections, and different indexes over the same collection. Thus, a program (`makeIndex`) is provided to do this task. It can choose among different stopword lists (previously inserted into the system) and use (if desired) Porter's stemming algorithm. Also, it is able to detect those situations explained in section 2.4.1 and create the corresponding virtual units.

Moreover, as stated by Baeza and Ribeiro [8], an IR system is characterized by its *weighting function*, w_{ij}, which assigns a weight to the term i in the document j. In the model, several valid weighting schemes could exist because of its experimental nature. As a consequence, in Garnata, indexing does not compute the weights (setting all of them to be zero). Instead of that, it includes the possibility to calculate weights (following a certain weighting scheme) for previously built indexes without inserting into them, and store them in files, the so-called *weight files*. So, records of that precomputed weight files are kept in order to provide a fast way to insert one into the index itself in order to retrieve with it. To achieve these two tasks two separated executables (`makeWeightFile` and `insertWeightFile`) are provided.

2.8.1.2 Physical level and data structures

Because indexes are only built few times, there is no need for using a very fast indexing algorithm. In Garnata, it is emphasized querying time over indexing time, even storing some redundant information. Nevertheless, we shall also describe the structures associated to indexing.

To store textual information (terms and identifiers of the final units where they appear), inverted indexes [104] are used. While the lexicon is kept entirely in memory (both while indexing and querying), the list of occurrences is read from disk. Another file is used to write the list of relative positions of each term inside a unit in order to answer queries containing proximity operators or phrases (although in the current stage of Garnata, they are not used to formulate a query).

A direct access file is used to maintain information about the structural units, except for the XPath routes, which are stored separately. Other files keep relations among units, being accessible with two disk reads only[1]. So a large file contains data of each unit itself (identifier, tag, container, position, ...) and besides, Garnata easily manages the following relationships with two disk accesses (see fig. 2.7 for more details):

[1] An access to this information requires two disk reads: the first in the direct access file (to retrieve the address) and the second in the proper data file (using retrieved address). This approach is similar to how a pointer works in programming languages.

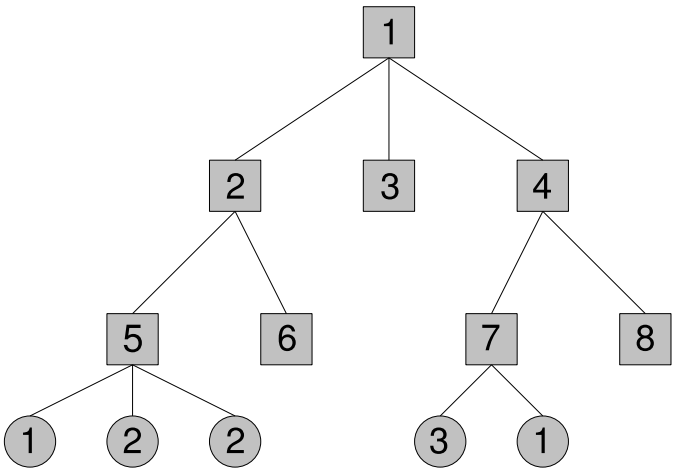


Figure 2.7: Relations between terms (circles) and units (squares).

- Given a non-final unit, returning the list of identifiers of the units that it contains. For unit 1 in fig. 2.7, it returns $(2, 3, 4)$.
- Given a final unit, returning the container unit and, recursively, all the containers until a root unit is found. For unit 5 in fig. 2.7, it returns $(2, 1)$.
- Given a final unit, returning the list of contained terms (known as the *direct index*). For unit 5 in fig. 2.7, it returns $(1, 2)$.

Although the indexing subsystem of Garnata is not aimed to any particular IR model, all of the previous operations are specifically designed for the CID model. Nowadays, this subsystem does not implement any type of file compression.

2.8.2 Querying subsystem

Querying subsystem is the most critical part of an IR system. In our case, structures are built at indexing time to reduce at maximum the amount of disk accesses while processing a query, in order to save time and give a short response

time. The algorithm for achieving this task comprises the following steps (not necessarily in this order):

1. Query is parsed, and occurrences of the component terms are retrieved from disk.

2. For each occurrence, implied final units are read into memory (if not already there).

3. For each final unit, its descendants are read into memory (if not already there).

4. Propagation is carried out, units are sorted by its relevance degree, and the result is returned.

The first big bottleneck to be minimized is due to the reading from disk of the unit objects (containing information about each unit). Garnata will keep two unit caches in memory: the first one, containing final units, and the second one, containing complex units. Both will be *static caches*, meaning that they will not change the unit stored in each cache slot. Cache is accessed doing a hash function-like scheme[1], so for each cache slot, there will be several candidates (those identifiers being the hash inverse of the slot identifier).

For the final units cache, in each slot, the unit containing greater number of terms (among the candidates) will be stored. For the complex units cache, in each slot, the unit containing more final units will be stored. These two heuristics have shown very good performance in different experiments.

[1]Identifier *mod* N, with N the size of the cache.

Part III

Methodological Contributions

Chapter 3

Improving the context-based influence diagram model for XML retrieval

3.1 Introduction

Since its beginnings, Garnata has suffered several modifications to improve its performance and broaden its field of action. So, several of these improvements have been one of the objectives of this thesis and, in some cases, they were the first steps to achieve the other contributions of this work.

Then, our different participations at INEX have permitted us to check the performance of the improvements since 2007. Thus, this chapter is focused on showing the modifications in Garnata and the results of the experimentation in different collections from INEX.

Firstly, we have adapted Garnata to cope with the three tasks proposed in INEX 2007 and following workshops, namely *focused, relevant in context* and *best in context* (section 3.2) because, for each query, Garnata generates a list of document parts or structural units, sorted by relevance value (expected utility), as the output. So, this output is compatible with the thorough task used in previous editions but not with the three *adhoc* tasks from INEX 2007.

Secondly, there are different parameters in Garnata that need to be fixed in order to use them. These parameters are the weights $w(B, S)$, $w(U, S)$, and the

utilities $v(r_i^+, u_i, u_{hi}(U_i))$ as we saw in chapter 2. In the case of the weights, it would be important to consider in its computation the importance of the unit U inside another unit S.

As we know, all the units from a document do not have the same importance, for instance, the terms from any title of a document or section, or terms in bold are more informative than the terms from other paragraphs of the document. On the other hand, the utility of a unit indicates whether this kind of unit is good from a retrieval point of view or not.

For example, if we are looking for information about a topic we shall be more interested in those units having more information about it. So, they correspond to the units containing more text (or terms) like a whole section or a paragraph instead of titles or terms in bold whose content is very limited for the users' need.

Following these ideas, we have proposed a modification of the model to redefine and compute them in a different way in section 3.3. Specifically, the computation of the utilities has suffered another modification by defining the utility in a different manner (section 3.4), in such a way that those components that do not contain most of the query terms are penalized more heavily.

By defining a parametric model, it is possible to adjust the degree of utility to make the system behaves more similarly to a strict AND (if not all or almost all the query terms are in the considered component, this one will be scarcely relevant) or to a less strict AND. For both kinds of improvements, we show some experimental results in the different INEX workshops. Lastly, the section 3.5 consists of some conclusions and future research for all these changes.

3.2 Adapting Garnata to the INEX tasks

To cope with the *adhoc* INEX tasks from 2007, *focused*, *relevant in context* and *best in context*, we shall use Garnata but after we filter its output in a way which depends on the type of task:

- **Focused task**:
  Assumption: Users want to see as much relevant text as possible with as little irrelevant text as possible.

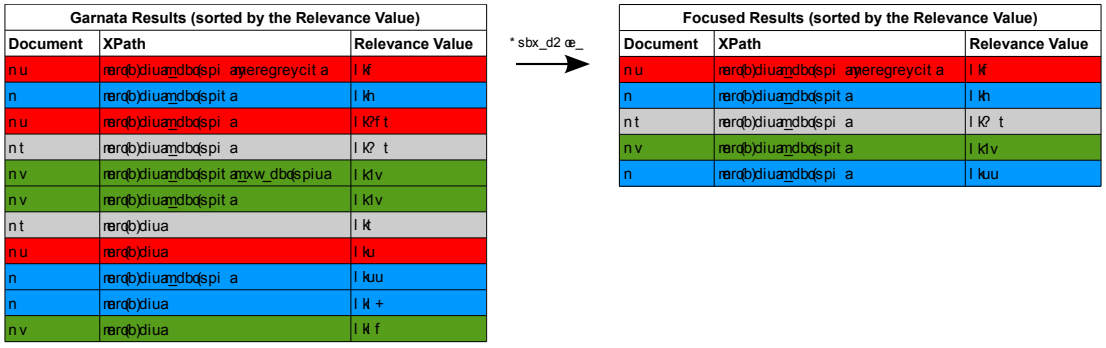


Figure 3.1: Example of "focused task".

Then, the output must be an ordered list of structural units where overlapping has been eliminated. So, we must supply some criterion to decide, when we find two overlapping units in the output generated by Garnata, which one to preserve in the final output. The criterion we have used is to keep the unit having the greatest relevance value and, in case of tie, we keep the most general unit (the one containing a larger amount of text). We can see an example in fig. 3.1. From D1, we only keep the `/article[1]/section[2]/paragraph[3]` unit because the rest of structural units from D1 are overlapped and it has the greatest relevance value. In this sense, we keep `/article[1]/section[2]` unit from D3. From D2, we keep `/article[1]/section[3]` and `/article[1]/section[2]` units because they are not overlapped and they have the greatest relevance values. Lastly, we find a tie between the `/article[1]/section[3]/subsection[1]` and `/article[1]/section[3]` overlapped units from D4, but we only keep the second unit because it is more general. Then, the final ranking is composed of these units sorted by their relevance values as we can see in the right side of the figure.

- **In context tasks**:

  Assumption: Users consider all relevant documents to be equally useful answers.

– **Relevant in Context task**: In this case the output must be an ordered list of documents and, for each document, a set of non-overlapping structural units, representing the relevant text within the document (i.e., a list of non-overlapping units clustered by document). Therefore, we have to filter the output of Garnata using two criteria: how to select the non-overlapping units for each document, and how to rank the documents. To manage overlapping units we use the same criterion considered for the focused task. To rank the documents, we have considered three criteria to assign a relevance value to the entire document. The relevance value of a document is equal to

  1. The maximum relevance value of its units.
  2. The relevance value of the "`/article[1]`" unit corresponding to the unit which represents the whole document.
  3. The sum of the relevance values of all its units.

  After some experimentation, the first criterion was chosen as the best alternative to rank the documents.

  We can see an example of this task in fig. 3.2. Following the three criteria to rank the documents, the rankings from the example are the following ones: For the first criterion, the ranking is D1-D2-D3-D4 corresponding to the order of their greatest relevance value units which are `/article[1]/section[2]/paragraph[3]` from D1, `/article[1]/section[3]` from D2, `/article[1]/section[2]` from D3 and `/article[1]/section[3]/subsection[1]` from D4. For the second criterion, the ranking is D3-D1-D2-D4 corresponding to the order of their `/article[1]` unit values. For the last criterion, the ranking is D1-D4-D3-D2 corresponding to the order of the sum values of the relevance values of all the units from each document.

  For instance, if we consider D1, its value is equal to the sum of the relevance values of `/article[1]/section[2]/paragraph[3]`, `/article[1]/section[2]` and `/article[1]` units from D1. To rank the units within a document, we can see the example of the focused task because the criterion is the same.

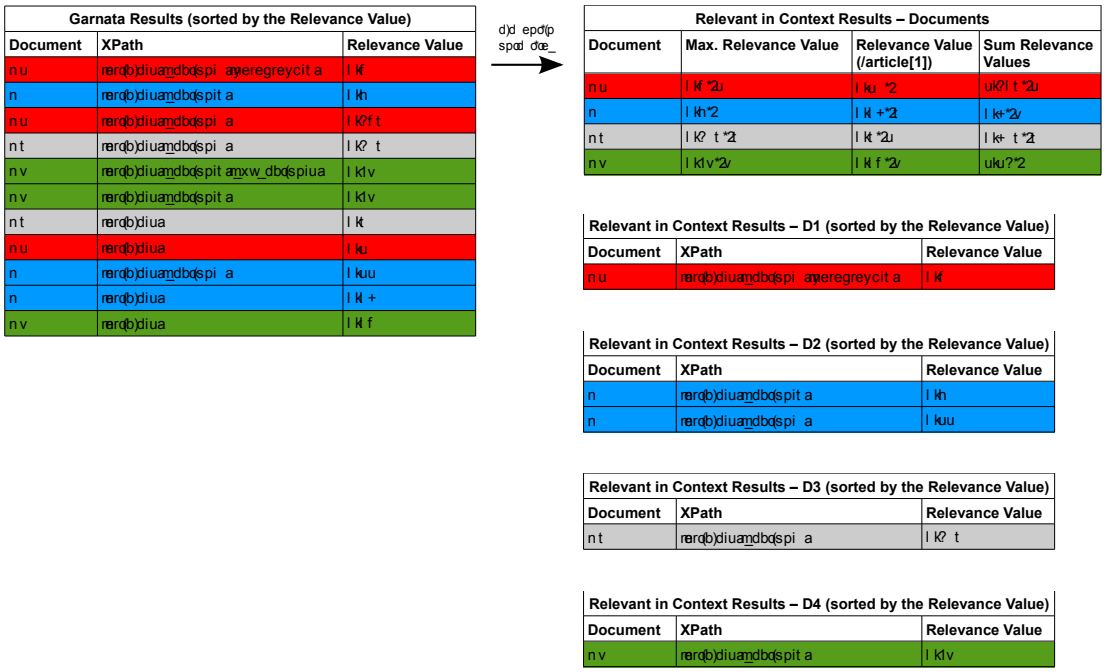


Figure 3.2: Example of "relevant in context" task (ranking for each criterion between brackets).

– **Best in Context task**: The output must be an ordered list composed of a single unit per document. This single document part should correspond to the best entry point for starting to read the relevant text in the document. Therefore, we have to provide a criterion to select one structural unit for each document and another to rank the documents/selected units. This last criterion is the same considered in the relevant in context task (the maximum relevance value of its units). Regarding the way of selecting one unit per document, the idea is to choose some type of *centroid* structural unit: for each unit U_i we compute the sum of the distances from U_i to each of the other units U_j in the document. The distance between U_i and U_j is measured as the number of links in the path between units U_i and U_j in the XML tree times the relevance value of unit U_j; then we select the unit having minimum sum of distances. In this way we try to select a unit which is nearest to the units having high relevance values.

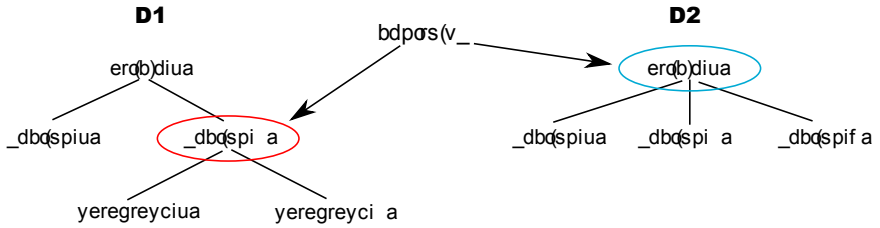
  As we can see from example in fig. 3.3, the centroids from D1 and D2 are `/article[1]/section[2]` and `/article[1]` units respectively because they have the minimum sum of distances. Afterwards, the final ranking list is composed of these two units and `/article[1]/section[2]` unit from D3 (it has the minimum sum of distances from D3) following the order of their relevance values; therefore, the ranking is `/article[1]/section[2]` unit from D1 (0.683), `/article[1]/section[2]` unit from D3 (0.623) and `/article[1]` unit from D2 (0.3).

3.3 Parametrizing Garnata

3.3.1 Recomputing weights depending on the type of unit

We have modified the way we compute the weights of the units (U) included in a complex unit (S), $w(U, S)$, in order to also take into account, not only the proportion of the content of S which is due to U, but also some measure of the

Garnata Results (sorted by the Relevance Value)		
Document	**XPath**	**Relevance Value**
n u	rerq(b)diua_mdbq(spi ayeregreyciua	t lk
n	rerq(b)diua_mdbq(spif a	t lh
n u	rerq(b)diua_mdbq(spi a	t lxkf
n f	rerq(b)diua_mdbq(spi a	t lx f
n u	rerq(b)diua	t lw?
n	rerq(b)diua	t lf
n u	rerq(b)diua_mdbq(spiua	t l k
n	rerq(b)diua_mdbq(spi a	t l x
n f	rerq(b)diua	t l ?
n u	rerq(b)diua_mdbq(spi ayeregreyci a	t l
n	rerq(b)diua_mdbq(spiua	t luk



D1	
XPath	**Sum of weighted distances**
erq(b)diua	u t l k+u t lxkf + t lk+ t l *2*f lt t f
_dbq(spiua	u t lw?+ t lxkf +f t lk+f t l *2*?l xx
_dbq(spi a	u t lw?+ t l k+u t lk+u t l *2* lu
yeregreyciua	t lw?+f t l k+u t lxkf + t l *2*f lt ?f
yeregreyci a	t lw?+f t l k+u t lxkf + t lk*2*?l t f

D2	
XPath	**Sum of weighted distances**
erq(b)diua	u t luk+u t l x+u t lh*2*ulu?
_dbq(spiua	u t lf + t l x+ t lh*2* l
_dbq(spi a	u t lf + t luk+ t lh*2* lt x
_dbq(spif a	u t lf + t l x+ t luk*2*uluk

Figure 3.3: Centroids of the documents for the “best in context” task.

importance of the type (tag) of unit U within S. For example, in the INEX 2007 dataset, the terms contained in a `collectionlink` (generally proper nouns and relevant concepts) or `emph2` should be quantified higher than terms outside those units. Units labeled with `title` are also very informative, but units with `template` are not.

So, we call $I(U)$ the importance of the unit U, which depends on the type of tag associated to U. These values constitute a global set of free parameters, specified at indexing time which are set using expert knowledge about the semantics of the tags. The new weights $nw(U,S)$, are then computed from the old ones (see eq. 2.4 in chapter 2) in the following way:

$$nw(U,S) = \frac{I(U)\cdot w(U,S)}{\sum_{U'\in Pa(S)} I(U')\cdot w(U',S)}. \tag{3.1}$$

We show in table 3.1 three different examples of importance schemes used in the official runs of INEX 2007. Unspecified importance values are set to 1 (notice that by setting $I(U)=1, \forall U \in \mathcal{U}$, we get the basic weighting scheme explained in chapter 2). As we commented before, these schemes remark the importance of the tags containing representative terms like `name`, `title`, `emph2`, etc. The difference between them is found in the importance value. The measures of W11 are, in general, higher than the ones of W8 increasing the importance of those units, but they are even more significant in W15. However, the importance of the tags containing non-informative terms (`conversionwarning`, `languagelink` and `template`) is 0 in the three schemes.

3.3.2 Recomputing utilities depending on the type of unit

The formula of the utility values for a unit U (see eq. 2.5 in chapter 2) is computed by considering another factor called relative utility value, $RU(U)$, which depends only on the type of tag associated to that unit, so that:

$$v(r_i^+, u_i, u_{hi(U_i)}) = nidf_Q(U_i)\cdot v(u_i, u_{hi(U_i)})\cdot RU(U_i). \tag{3.2}$$

It should be noticed that this value $RU(U)$ is different from the importance $I(U)$: a type of unit may be considered very important to contribute to the

Tag	Weight 8 (w8)	Weight 11 (w11)	Weight 15 (w15)
name	20	100	200
title	20	50	50
caption	10	10	30
collectionlink	10	10	30
emph2	10	30	30
emph3	10	30	30
conversionwarning	0	0	0
languagelink	0	0	0
template	0	0	0

Table 3.1: Importance of the different types of units used in the official runs.

relevance degree of the unit containing it and, at the same time, is considered not very useful to retrieve this type of unit itself.

For example, this may be the case of units having the tag `title`: in general a title alone may be not very useful for a user as the answer to a query. Probably, the user would prefer to get the content of the structural unit having this title. However, terms in a title tend to be highly representative of the content of a document part, so that the importance of the title should be greater than the importance derived simply of the proportion of text that the title contains (which will be quite low).

An example of sets of utility values used in the official runs of INEX 2007 is displayed in table 3.2. In all the cases, the default value for the non-listed units is 1.0. We have also considered the case where all the relative utility values are set to 1.0 (which is equivalent to not using relative utilities at all).

The three sets have low utility values for `title`, `name` and `collectionlink` (in u1 and u3) units because their content in itself is not considered an informative answer for a query. However, in u1 and u3 `article`, `section`, `p` and `body` units take higher values because they are the units containing enough information from the document to answer most of the questions. It is important to mention that the greatest value is given to the `article` unit due to the fact that documents from INEX, in general, are really short, so the whole document can be the perfect answer to many of the queries from INEX.

Tag	Utility 1 (u1)	Utility 2 (u2)	Utility 3 (u3)
conversionwarning	0	0	0
name	0.75	0.75	0.85
title	0.75	0.75	0.85
collectionlink	0.75	1.5	0.75
languagelink	0	0	0
article	2	2.5	2.5
section	1.5	1	1.25
p	1.5	1	1.5
body	1.5	1	2
emph2	1	1.5	1
emph3	1	1.5	1

Table 3.2: Relative utility values of the different types of units used in the official runs.

To check another scheme, u2 contains great utility values in `collectionlink`, `emph2` and `emph3` units, which represents more utility to the terms in bold and links, decreasing the utility values of the tags commented before, except for `article` unit which is always the most representative tag. As well as the importance values $I(U)$, `conversionwarning` and `languagelink` utility values are 0 given the fact that they do not contain interesting information.

3.3.3 Experimental evaluation

In our participation in INEX 2007, we obtained the results in the three tasks displayed in tables 3.3, 3.4 and 3.5, using the combinations of weight and utility configurations displayed in tables 3.1 and 3.2.

Besides, the utility values used in these experiments are $v^{--} = v^{-+} = v^{++} = 0, v^{+-} = 1$ where v^{--} represents the utility of retrieving a non-relevant unit given its non-relevant child , v^{-+} represents the utility of retrieving a non-relevant unit given its relevant child, v^{++} represents the utility of retrieving a relevant unit given its relevant child, v^{+-} represents the utility of retrieving a relevant unit given its non-relevant child.

Then, the main objective of these values is to retrieve the most general unit from the list of retrieved units.

Weight	Utility	Ranking
w8	u3	62/79
w15	u2	70/79
w15	none	71/79

Table 3.3: Results for the "focused" task.

Weight	Utility	Ranking
w15	u3	44/66
w8	u3	45/66
w11	u1	47/66

Table 3.4: Results for the "relevant in context" task.

As we can see in these results, the configuration of utilities u3 is the most appropriate to get the best results in the different tasks, although we have not found a specific configuration of weights that obtain the best results.

3.4 Parametric utility model

As it can be observed from eq. 2.6 in chapter 2, the utility or exhaustivity of a structural unit U with respect to a query Q grows linearly with the number of query terms appearing in U (reaching a maximum equal to 1 when all the terms of the query appear in the unit).

In our experience with the system in different applications, described by Campos et al. in [29, 26], we have observed that this linear growing, when combined with the probabilities computed from the Bayesian network (which measure specificity), can cause that small structural units, which only match with a fraction of the query terms, become more relevant than greater structural units that contain

Weight	Utility	Ranking
w8	u3	40/71
w15	none	45/71
w15	u2	47/71

Table 3.5: Results for the "best in context" task.

more terms from the query. In many cases this behaviour is not the expected one, because probably a user who employs several terms to express his/her query is expecting to find most of these terms in the structural units obtained as the answer of the system to this query.

For instance, if we introduce the query "*agricultural benefits in Granada*", we expect to get all the units containing the three main terms (*agricultural*, *benefits* and *Granada*) because they have more probabilities to talk about this query than the rest of units containing one or two query terms. These units can talk about agricultural benefits in other cities or agriculture in Granada, for example. In this way, we believe that it is interesting to define other utility models which give more importance (in a non-linear way) to the appearance of most of the terms in the query.

In this section we propose a parametric non-linear utility model that, as the parameter grows, the more terms from the query must be contained in a structural unit in order to get a high utility value for this unit. A way of obtaining this behaviour is through the use of the following transformation:

$$nidf_{Q,n}(U) = nidf_Q(U)\frac{e^{(nidf_Q(U))^n} - 1}{e - 1}. \tag{3.3}$$

In this way, when $n = 0$ we have $nidf_{Q,0}(U) = nidf_Q(U)$, that is to say, we reproduce the original model, and the greater the value of the integer parameter n, we obtain a behaviour more similar to a strict AND operator. In fig. 3.4 we can observe several plots of the function $x\frac{e^{x^n}-1}{e-1}$ for different values of n.

3.4.1 Experimental evaluation

Firstly, the evaluation of this new parametric utility model has been done with the collection used in INEX 2007. In terms of queries, there were 99 queries with their relevance assessments in this edition.

In fig. 3.5, 3.6 and 3.7, we can see the evolution of the measures of retrieval effectiveness for the different values of n.

In tables 3.6, 3.7 and 3.8, we can observe the relative positions, we would obtain, comparing our results for the different measures with the results of INEX 2007.

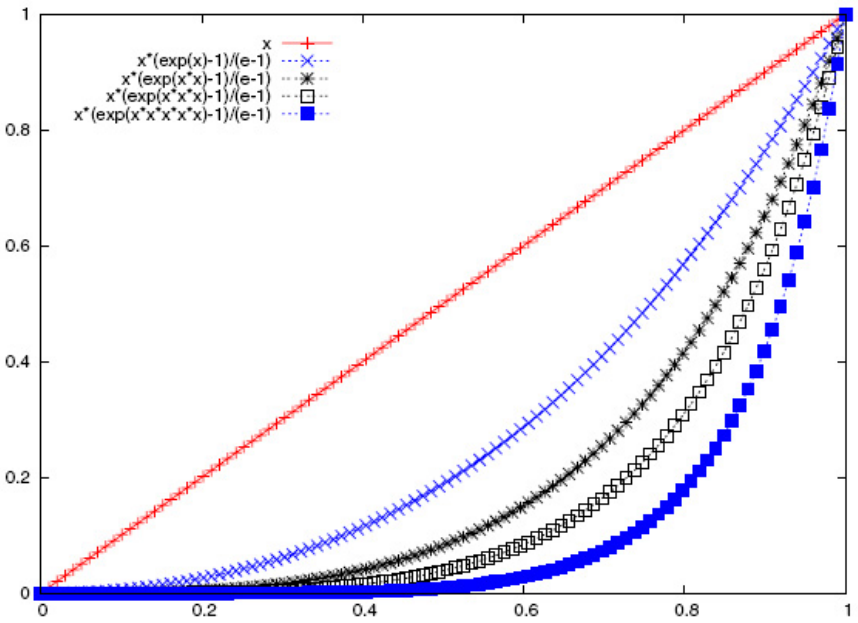


Figure 3.4: Function $x\frac{e^{x^n}-1}{e-1}$, for $n = 0, 1, 2, 3, 5$.

n	0	1	2	3	5
iP[0.0]	61/79	51/79	53/79	47/79	47/79
iP[0.01]	67/79	53/79	51/79	41/79	42/79
iP[0.05]	68/79	52/79	48/79	45/79	53/79
iP[0.1]	69/79	50/79	40/79	39/79	39/79
MAiP	68/79	49/79	37/79	33/79	33/79

Table 3.6: Relative positions in the INEX ranking for the different measures in the "focused task", for $n = 0, 1, 2, 3, 5$.

n	0	1	2	3	5
gP[5]	51/66	32/66	34/66	34/66	34/66
gP[10]	51/66	39/66	37/66	32/66	29/66
gP[25]	48/66	33/66	31/66	29/66	28/66
gP[50]	49/66	36/66	32/66	29/66	31/66
MAgP	44/66	40/66	37/66	31/66	31/66

Table 3.7: Relative positions in the INEX ranking for the different measures in the "relevant in context task", for $n = 0, 1, 2, 3, 5$.

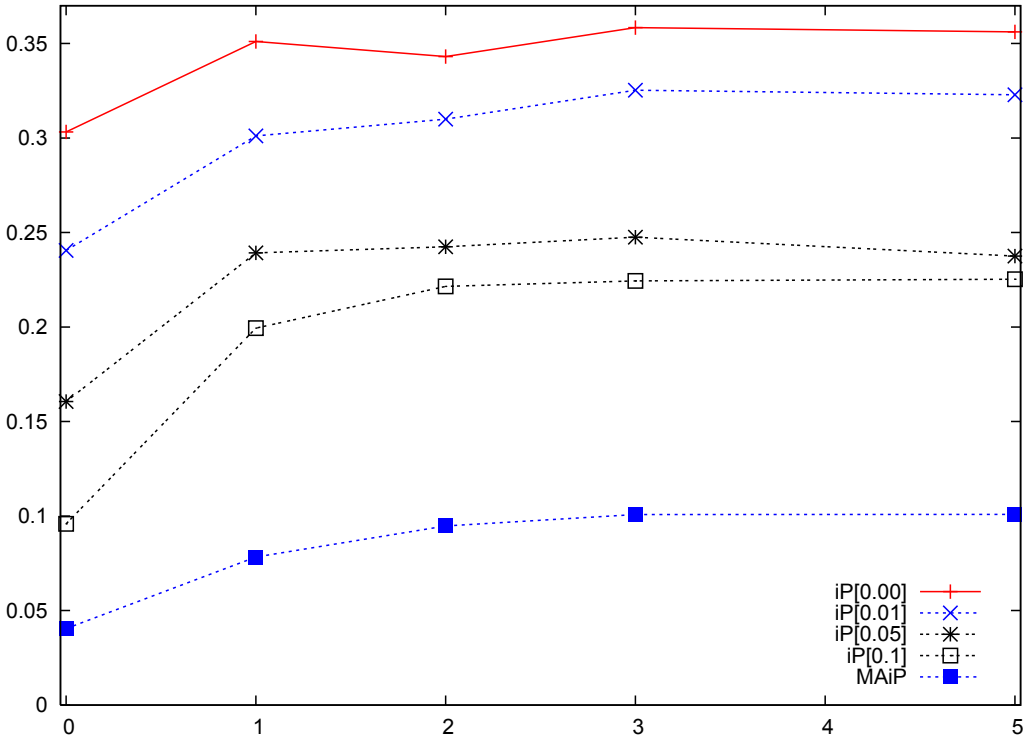


Figure 3.5: Results of the measures of retrieval effectiveness in the "focused" task, for $n = 0, 1, 2, 3, 5$.

n	0	1	2	3	5
gP[5]	43/71	14/71	12/71	12/71	12/71
gP[10]	44/71	15/71	10/71	13/71	10/71
gP[25]	48/71	34/71	23/71	13/71	17/71
gP[50]	47/71	31/71	24/71	18/71	17/71
MAgP	45/71	29/71	24/71	21/71	21/71

Table 3.8: Relative positions in the INEX ranking for the different measures in the "best in context task", for $n = 0, 1, 2, 3, 5$.

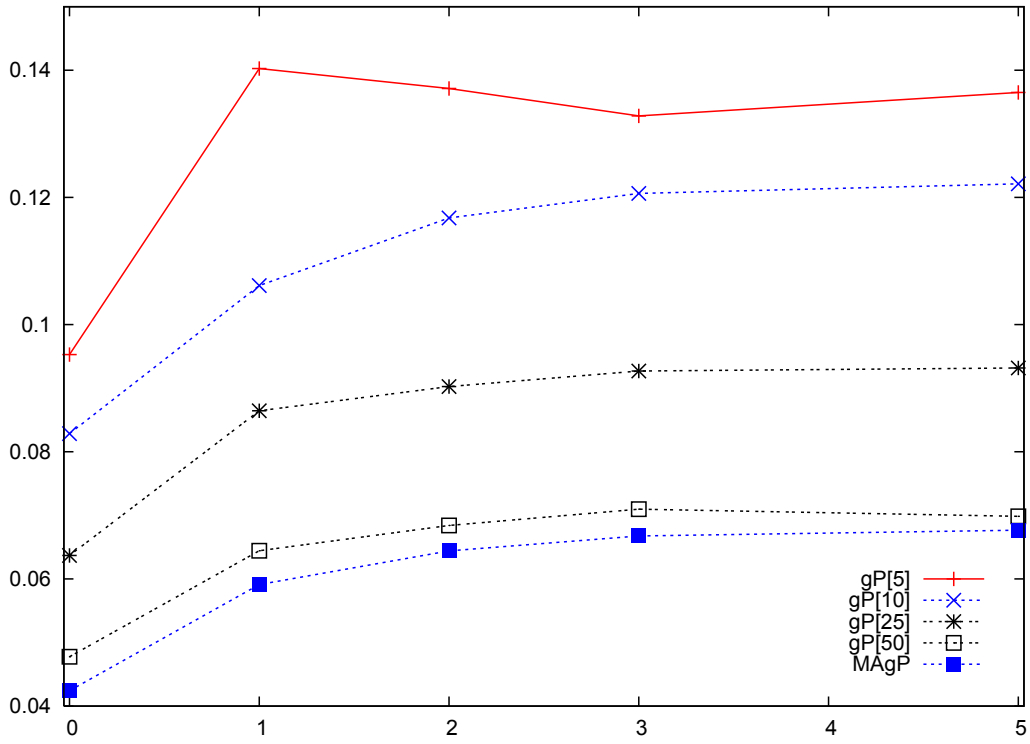


Figure 3.6: Results of the measures of retrieval effectiveness in the "relevant in context" task, for $n = 0, 1, 2, 3, 5$.

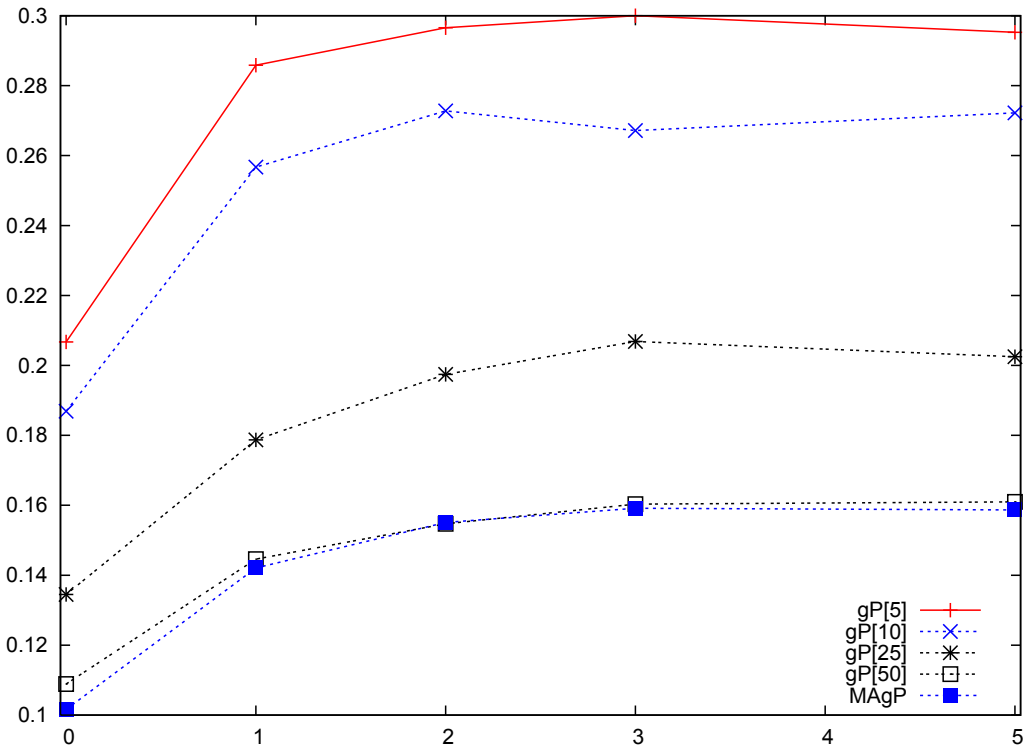


Figure 3.7: Results of the measures of retrieval effectiveness in the "best in context" task, for $n = 0, 1, 2, 3, 5$.

Position	Value	Sub-task	Weight	Utility	n	$v^{++}, v^{+-}, v^{-+}, v^{--}$
52	0.468856	Foc	w8	u3	5	1,1,1,0
53	0.467071	Foc	w8	u3	3	1,1,1,0
54	0.448733	Foc	w15	u3	5	1,1,1,0
25	0.158177	RIC	w8	u3	5	1,1,1,0
26	0.158177	RIC	w8	u3	5	0,1,0,0
27	0.152320	RIC	w8	u3	3	1,1,1,0
18	0.146799	BIC	w8	u3	5	0,1,0,0
19	0.146536	BIC	w8	u3	3	0,1,0,0
22	0.138141	BIC	w15	u3	3	0,1,0,0

Table 3.9: Runs submitted to the INEX 2008 *adhoc* tasks and positions in the rankings. (Foc: Focused, RIC: Relevant in context, BIC: Best in context).

In conclusion, if we observe all the figures and tables, we can appreciate an important improvement in the parametric utility model using $n > 0$, getting the best results when n takes the values 3 and 5.

Therefore, we decided to participate with this model in INEX 2008 edition, submitting nine runs in the *adhoc* track (content only). More specifically, three in each of the focused, relevant in context and best in context sub-tasks. Table 3.9 shows the positions in the ranking according to the official evaluation measures (MAgP for best in context and relevant in context, and iP [0.01] for focused), the sub-task, the weight and utility configurations, and finally the utility values.

With respect to the parameters, we have used the weight schemes 8 and 15 (w8 and w15), and utility set 3 (u3), with the first values presented in table 3.1 and in table 3.2 the second ones.

We have experimented with the two best values of the parameter n in eq. 3.3, 3 and 5, for INEX 2007. Finally, the last four values of the table corresponds to the values of each of the four configurations of the component of the utility function independent on the involved unit (see section 2.5 of chapter 2).

In order to better determine the improvement obtained by the new utility model presented in this section, we have run an experiment without using the transformation presented in eq. 3.3, but applying instead the original eq. 2.6, $nidf_Q(U)$. Table 3.10 shows the values of the official evaluation measures with the old utility model used in previous editions of INEX (first column), INEX 2008 with the new model (second column) and the percentage of change (third column). As noticed, the percentages of change are generally quite large, and this

With $nidf_Q(U)$	With $nidf_{Q,n}(U)$	%Change	Sub-tasks	Weight	Utility	n	$v^{++}, v^{+-}, v^{-+}, v^{--}$
0.366249	0.468856	28.01	Foc	w8	u3	5	1,1,1,0
0.366249	0.467071	27.53	Foc	w8	u3	3	1,1,1,0
0.341804	0.448733	31.28	Foc	w15	u3	5	1,1,1,0
0.083034	0.158177	90.50	RIC	w8	u3	5	1,1,1,0
0.067706	0.158177	133.62	RIC	w8	u3	5	0,1,0,0
0.083034	0.152320	83.44	RIC	w8	u3	3	1,1,1,0
0.075842	0.146799	93.56	BIC	w8	u3	5	0,1,0,0
0.075842	0.146536	93.21	BIC	w8	u3	3	0,1,0,0
0.078910	0.138141	75.06	BIC	w15	u3	3	0,1,0,0

Table 3.10: Comparison between runs with and without applying the transformation in eq. 3.3. (Foc: Focused, RIC: Relevant in context, BIC: Best in context).

fact confirms our initial hypothesis that the new transformation could improve the results.

We have carried out another series of experiments, motivated by the following fact: we realized that among the systems obtaining the best results in the official competition at INEX 2008, as we can see in [43], there are many systems that do not return any possible structural unit as a result but only some of them, typically only content bearing elements like section, paragraphs or the complete article.

In contrast, our official runs retrieved almost any elements, and this may be a source of poor behaviour specially when removing overlapping elements. So, we have repeated our official experiments but filtering the results in order to retrieve only article, or only article, body, section and paragraph elements. This can be easily done by using an utility file giving weight zero to all the structural units except the selected ones (with weight equal to one). The results of these experiments are displayed in table 3.11.

We can observe that this strategy of retrieving only the more general elements is useful for the focused and relevant in context tasks, where we would obtain better positions in the ranking (going from percentiles 85-88 to 75-85 in focused and from 62-67 to 50-60 in relevant in context, when using the four elements selected). However, the results are slightly worse for the best in context task (going from percentiles 51-63 to 57-68) in the case of using the four elements but better when using only the article element. These results point out that the

article+section+...		only article						
Position	Value	Position	Value	Sub-task	Weight	Utility	n	$v^{++}, v^{+-}, v^{-+}, v^{--}$
48	0.517808	52	0.482262	Foc	w8	u3	5	1,1,1,0
46	0.524948	52	0.478478	Foc	w8	u3	3	1,1,1,0
52	0.474641	54	0.455649	Foc	w15	u3	5	1,1,1,0
20	0.171119	27	0.157455	RIC	w8	u3	5	1,1,1,0
24	0.164420	27	0.157455	RIC	w8	u3	5	0,1,0,0
22	0.168308	27	0.155347	RIC	w8	u3	3	1,1,1,0
20	0.146501	14	0.168893	BIC	w8	u3	5	0,1,0,0
22	0.140705	14	0.167468	BIC	w8	u3	3	0,1,0,0
24	0.131170	18	0.148391	BIC	w15	u3	3	0,1,0,0

Table 3.11: Runs retrieving only content-bearing elements and positions in the rankings. (Foc: Focused, RIC: Relevant in context, BIC: Best in context).

choice of the structural elements to be retrieved has a non-negligible impact on the performance of an XML retrieval system.

3.5 Concluding remarks

In this chapter we have presented all the improvements we have implemented in our XML IR system, Garnata. Firstly, we have adapted Garnata to the three *adhoc* tasks of INEX: Focused, best in context and relevant in context. We have improved the way to use different weights and utilities and finally, we have created a new parametric utility model which gives more importance (in a non-linear way) to the appearance of most of the terms in the query.

Our last participation in the INEX 2008 edition in the *adhoc* tasks has demonstrated that these modifications considerably improve the results with respect to not using them.

With respect to the comparison of our results with the rest of participants, we could say that we are in the middle of the rankings, improving with respect to the previous editions of INEX.

Regarding future research in the context of INEX, we have to work in the improvement of the raw results of Garnata, as they are the base for the different sub-tasks, and in the filtering strategy used to remove overlapping elements because as we saw in the last experimentation of section 3.4.1, the retrieval of determined units like article or sections, instead of almost any element, could improve the results considerably.

Chapter 4

Content and Structure queries

4.1 Introduction

The inclusion of the structure of documents affects the design and implementation of the IR system in many ways, as we commented in chapter 2: the indexing, retrieval processes and the interactive way to introduce queries. In fact, querying by content and structure can only be achieved if the user can specify in the query *what* he or she is looking for, and *where* this should be located in the required documents. The "what" involves the specification of the content, while the "where" is related to the structure of the documents.

There are already many IR systems which are able to deal with structured documents and the series of INEX Workshop proceedings [39, 38, 36, 37, 40, 35] are an excellent source of information. It is, however, also true that in many cases these systems take as input non-structured queries involving only content (the so-called *content-only*, CO queries[1]), although their output may be any type of document component.

What we propose in this chapter is a general methodology to convert some of these systems into fully structured IR systems which are able to process structured queries involving both content and structure (the so-called CAS queries). Our starting point, therefore, is any information retrieval system which is capable of computing (given a non-structured query) a relevance value for each structural

[1] Queries made of terms or words and not containing any reference or restriction relative to structural elements in the document collection.

unit in a document, representing the posterior probability of relevance of that unit given the query[1].

In section 4.2 of this chapter, we explain the type of structured query being dealt with. Section 4.3 describes our proposal for transforming a partially structured IR system into a fully structured one, by allowing the system to manage structured queries. Section 4.4 reviews some of the existing approaches for managing Content and Structure queries, whereas the experimental evaluation of our proposal is explained in section 4.5. Finally, section 4.6 contains our concluding remarks and various proposals for future research.

4.2 Introduction to structured queries

XPath, *XML Path Language* [1], is a query language for selecting nodes from an XML document as we commented in chapter 1. Then, *XPath* language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting the required nodes.

With respect to the two different views of the XML content: data-centric and text-centric XML. In this chapter, we are focussed on the text-centric XML view because the aim of text-centric XML retrieval is to develop methods to find correspondence between the text of the query and the text of the XML documents considering the structural restrictions too. Therefore, this is the context in which the chapter is set.

In order to allow queries combining content and structure (Content and Structure queries, CAS, in INEX terminology) to be specified, the NEXI language [97] was designed. It is a simplified XPath containing only the descendant operator (//) in a tag path and also an extended XPath containing the *about* function. NEXI has been used by INEX since 2004.

The type of structured CAS query considered by NEXI can take two possible forms:

- $//C[D]$: Returns C units that fulfill the condition D.

[1]The restriction to probabilistic systems is more formal than practical, because the proposed method will work with any system returning relevance values in the $[0, 1]$ interval. Otherwise, any type of normalization could be used.

- $//A[B]//C[D]$: Returns C descendants of A where A fulfills the condition B and C fulfills the condition D.

A and C are *paths* (sequences of elements or structural units), specifying structural restrictions, whereas B and D are *filters*, which specify content restrictions, and // is the descendant operator. C is the *target* path (the last structural unit in C is the one that we want to retrieve) and path A is the *context*. Each content restriction will include one or several *about* clauses, connected by either *and* or *or* operators; each *about* clause contains a text (a sequence of words or terms) together with a relative path, from the structural unit which is the container of the clause to the structural unit contained in it where this text should be located. The *about* clause is the IR counterpart of the classical *contains* clause used in XPath (which requires an exact matching between the textual content of the clause and a part of the text in the structural element being evaluated). However, *about* does not demand such a strict matching but states, vaguely, that a relevant element should satisfy the information need expressed by means of the text contained in the clause.

Example 1: Let us suppose that the hierarchical structure (e.g. the XML tree) of a document collection is the one displayed in fig. 4.1. An example of a NEXI-structured query is the following:
//A[about(.,text2) and about(.//F,text3) and about(.//J,text4)]//D[about(.,text1) and about(.//N,text5)].

What we want to retrieve with this query are D units which are contained within A units. The target D units should speak about *text1* and contain an N unit speaking about *text5*; the context A units should be about *text2* and also contain F and J units dealing with *text3* and *text4*, respectively (see fig. 4.2).

4.3 Managing structured queries

In this section, we shall explain how a structured CAS query of the type considered in section 4.2 can be managed using an IR system able to process only CO queries.

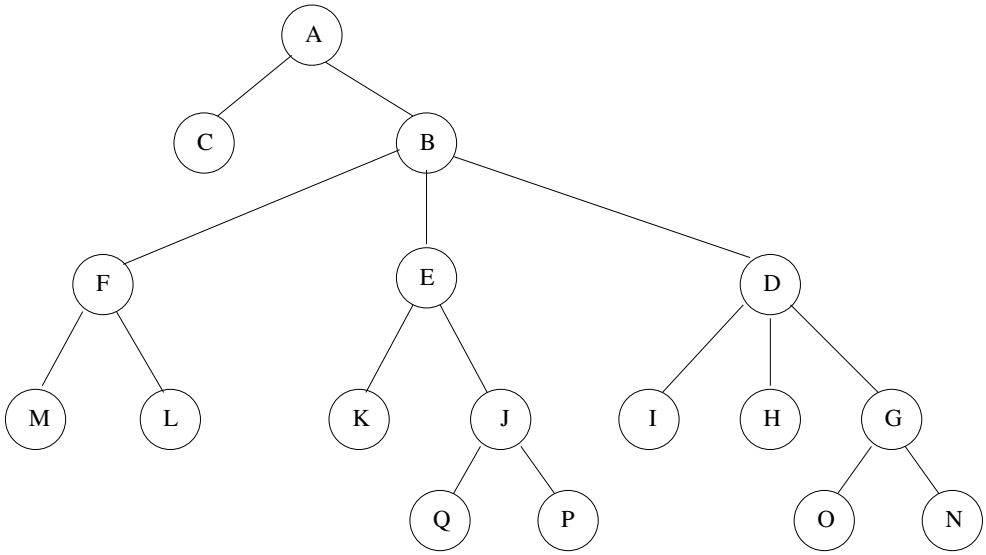


Figure 4.1: Tree structure of the documents for Example 1.

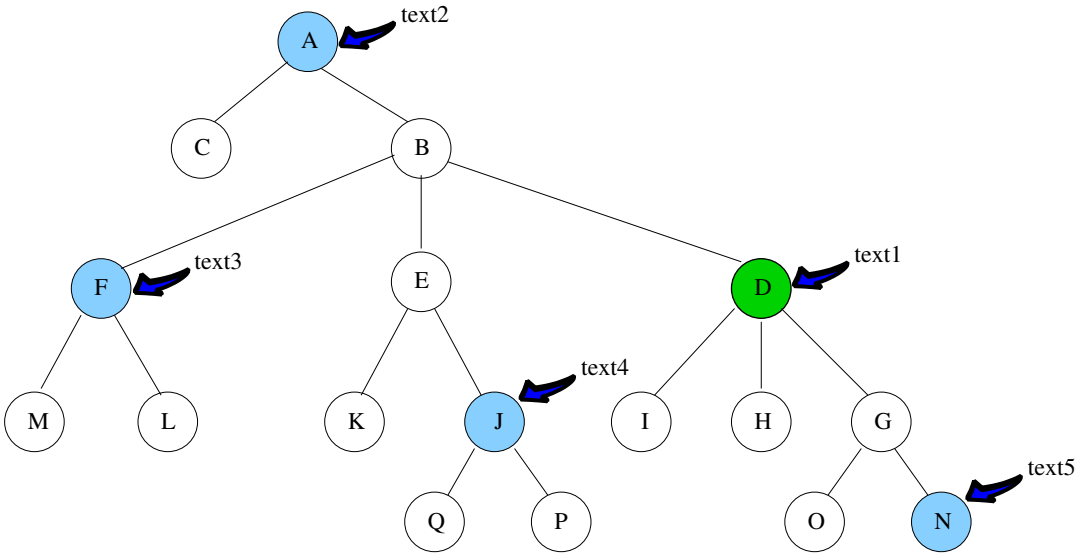


Figure 4.2: Structural units in the hierarchy where some pieces of text have been specified for Example 1.

Each about clause which is part of the structured query will give rise to a (non-structured) subquery. This subquery will be used to compute the posterior probabilities of the structural units specified within the clause, by postprocess-

ing the output of the base system (removing the units which do not satisfy the structural conditions specified in the about clause and in the path[1]).

Fig. 4.3 shows the modules required for a probabilistic retrieval system in order to deal with structured queries. A *NEXI Query Processor* extracts the content subqueries from the NEXI query (those which occur in the about clauses). These are passed to the XML retrieval system. It then runs a retrieval and obtains a ranking of relevant elements for each subquery. With these rankings plus the original NEXI query, the last module, the *Output Processor*, filters the results that do not satisfy the structural restrictions for each subquery, selects the objective elements and computes an RSV (relevance status value) for each element, returning a sorted list of units satisfying the initial query.

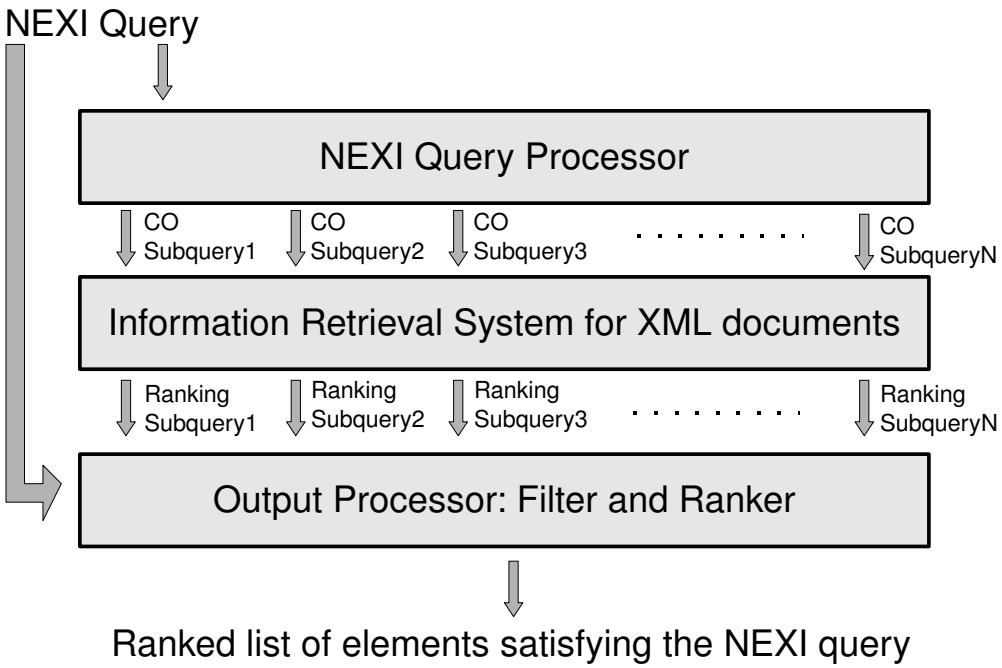


Figure 4.3: Architecture of a system for managing Content and Structure queries.

[1]This could also be done by modifying the base system, in order to compute *only* the probabilities of the structural units satisfying these structural conditions. This requires a more complex interaction with the base system but would possibly result in greater efficiency.

Let us consider the following example (simple query of type $//A[B]$):

Example 2:

Q = //chapter[about(.//title,text1) and about(.//section,text2)].

This query, in the context of a book collection, searches for chapter units with a title about *text1* and containing a section about *text2*. This is an example of the first, simpler type of query supported by NEXI.

In this example, the posterior probabilities computed for the subquery *text1* only focus on the title units which are contained in chapter units; the posterior probabilities for the subquery *text2* are also restricted to those section units contained within chapter units. In this way, structural units associated to paths such as, for example, */book/chapter/section*, */book/chapter/section/paragraph* or */book/chapter/author* would be discarded for the first subquery, whereas structural units associated to paths such as */book/chapter/section/paragraph*, */book/chapter/title* or */book/chapter/author* would also be eliminated for the second subquery.

Once we have discarded these units (thus keeping only the probabilities $p((//chapter//title)_l \mid text1)$[1] and $p((//chapter//section)_l \mid text2)$ in the example), we must compute the probability of the structural unit containing the about clause(s) (in the example this is the probability of the chapter units, $p((//chapter)_l \mid Q)$). This requires:

1. computing the aggregated probability of all the structural units of the same type associated to the corresponding unit (in the case of the container unit having more than one of these units) and then

2. combining these aggregated probabilities for the different structural units specified in the different about clauses.

In the example, we need to compute the aggregated probability of the section units (step 1), $p((//chapter)_l//section \mid text2)$ (because, in this case, a chapter

[1] In this context, an XPath reference such as $//chapter//title$ represents the set of all the structural units compatible with this reference, and $(//chapter//title)_l$ denotes an element in this set.

may contain more than one section), whereas the probability of the title unit, $p((//chapter)_l//title \mid text1)$, does not change, because a chapter will only have a single title[1]. The combined probability of section units and title units will then be associated to their container chapter units (step 2), $p((//chapter)_l \mid Q)$.

In order to satisfy the about clauses, it is sufficient for one section to be relevant (and also the corresponding title of the chapter), and it is not necessary for many or all of the sections of a chapter to be relevant. Therefore, to aggregate the probabilities of the structural units of the same type (when the container unit can have more than one of these units), it is appropriate to use a combination model representing a disjunction. However, to combine the probabilities of different types of units inside the container unit, this will depend on the type of connective being considered, either conjunction ("and") or disjunction ("or") (it will be a conjunction in the example). If a given chapter, i, contains exactly three sections and one title, the probability of the chapter would be obtained as

$$\begin{aligned} p((//chapter)_i \mid Q) &= p((//chapter)_i//title \mid text1) \text{ AND} \\ &\quad p((//chapter)_i//section \mid text2) \\ &= p(((//chapter)_i//title)_1 \mid text1) \text{ AND} \\ &\quad \big(p(((//chapter)_i//section)_1 \mid text2) \text{ OR} \\ &\quad p(((//chapter)_i//section)_2 \mid text2) \text{ OR} \\ &\quad p(((//chapter)_i//section)_3 \mid text2)\big), \end{aligned}$$

where AND and OR should be understood as operators for aggregating probabilities (instead of the boolean operators).

This process will compute the posterior probabilities of the structural units containing the about clauses specified in the query, for both the context (A) and the target (C), but these probabilities must also be combined to obtain the final probabilities of the target structural units. In the previous example this is not necessary because we only specified a target unit and not any context for this

[1] Although we would also need to aggregate the probabilities if section units may also contain title units, because the path /chapter/section/title also matches //chapter//title.

unit. Let us, however, examine another, more complex example (query of type $//A[B]//C[D]$):

Example 3:

Q=//chapter[about(.//title,text1) and about(.//section,text2)]//bibliography [about(.,text3)].

This query attempts to retrieve bibliography units containing *text3*, which must be in chapter units with a *text1*-related title and contain a section about *text2*. The bibliography units are the target and the chapter units are the context.

In this case, in addition to computing the posterior probabilities of chapter units, $p((//chapter)_l \mid B)$ as in the previous example, we would similarly compute the posterior probabilities of bibliography units in chapter units given the query $D = text3$, $p((//chapter//bibliography)_l \mid D)$. It is then necessary to combine $p((//chapter)_l \mid B)$ with $p((//chapter//bibliography)_l \mid D)$ to obtain the final relevance degree of each bibliography unit in relation to the structured query, $p((//chapter//bibliography)_l \mid Q)$. This combination clearly must also be conjunctive.

4.3.1 Combining probabilities

In the general case, let us consider a query Q =//A[B]//C[D], i.e. $Q = Q_A$ AND Q_C, where Q_A =//A[B], Q_C =//A//C[D], and

$$B = [\text{about}(.//A_1, \text{text}_{A1}) \text{ and/or} \ldots \text{and/or about}(.//A_n, \text{text}_{An})].$$

$$D = [\text{about}(.//C_1, \text{text}_{C1}) \text{ and/or} \ldots \text{and/or about}(.//C_m, \text{text}_{Cm})].$$

Since this query gives rise to $n + m$ subqueries, we use the base IR system to compute the probabilities $p(. \mid \text{text}_{Ai})$ and $p(. \mid \text{text}_{Cj})$, $i = 1, \ldots, n$, $j = 1, \ldots, m$, for all the structural units.

Firstly, we filter the results of these queries to focus on the structural units of interest, namely $p((//A//A_i)_l \mid \text{text}_{Ai})$ for each about clause in the context and $p((//A//C//C_j)_l \mid \text{text}_{Cj})$ for each about clause in the target.

Secondly, these probabilities are used to obtain the aggregated probabilities of the structural units of the same type, $p((//A)_l//A_i \mid \text{text}_{Ai})$ for the context and $p(((//A)_l//C)_h//C_j \mid \text{text}_{Cj})$ for the target, which are computed as

$$\begin{aligned} p((//A)_l//A_i \mid \text{text}_{Ai}) &= OR_k\big(p(((//A)_l//A_i)_k \mid \text{text}_{Ai})\big), \\ p(((//A)_l//C)_h//C_j \mid \text{text}_{Cj}) &= OR_k\big(p((((//A)_l//C)_h//C_j)_k \mid \text{text}_{Cj})\big). \end{aligned} \quad (4.1)$$

Thirdly, we combine these aggregated probabilities for the different structural units specified in the different about clauses, in order to obtain the probabilities for the content and the target, $p((//A)_l \mid Q_A)$ and $p(((//A)_l//C)_h \mid Q_C)$, respectively:

$$\begin{aligned} p((//A)_l \mid Q_A) &= \text{OP}_{i=1}^{n}\big(p((//A)_l//A_i \mid \text{text}_{Ai})\big), \\ p(((//A)_l//C)_h \mid Q_C) &= \text{OP}_{j=1}^{m}\big(p(((//A)_l//C)_h//C_j \mid \text{text}_{Cj})\big), \end{aligned} \quad (4.2)$$

where the operator OP may be either conjunction (AND) or disjunction (OR).

The final probability of the desired structural units, $p(((//A)_l//C)_h \mid Q)$, will be obtained as

$$p(((//A)_l//C)_h \mid Q) = \text{AND}\big(p((//A)_l \mid Q_A), p(((//A)_l//C)_h \mid Q_C)\big). \quad (4.3)$$

4.3.2 Probabilities of disjunctions and conjunctions

In order to manage disjunctive and conjunctive aggregations in a probabilistic setting, a simple and reasonable option is to use (noisy) OR and (noisy) AND gates. Let $U_1, \ldots, U_n$ be a set of structural units, and we want to compute the disjunction $\text{OR}_{i=1}^{n}(p(U_i))$ and the conjunction $\text{AND}_{i=1}^{n}(p(U_i))$.

4.3.2.1 Noisy-OR gates

A noisy OR gate is defined by means of [73]:

$$\text{OR}_{i=1}^{n}(p(U_i)) = 1 - \prod_{i=1}^{n}\left(1 - w_o(U_i)p(U_i)\right), \quad (4.4)$$

where $w_o(U_i)$, with $0 \leq w_o(U_i) \leq 1$, is a weight representing the probability that the disjunction is true if U_i alone is true, the other units U_j, $j \neq i$, being false. If $\forall i \; w_o(U_i) = 1$, we have a pure (non-noisy) OR gate, which in this case is equivalent to using the probabilistic sum t-conorm [86].

4.3.2.2 Noisy-AND gates

A noisy AND gate is defined by means of [73]:

$$\text{AND}_{i=1}^{n}(p(U_i)) = \prod_{i=1}^{n} \big(1 - w_a(U_i)(1 - p(U_i))\big), \tag{4.5}$$

where $w_a(U_i)$, with $0 \leq w_a(U_i) \leq 1$, in this case is the probability that the conjunction is false if U_i is false and all the other units U_j, $j \neq i$, are true. As before, if $\forall i \; w_a(U_i) = 1$, we have a pure (non-noisy) AND gate and the probability of the conjunction may be reduced to the product of the individual probabilities (which is equivalent to using the product t-norm [86]).

4.3.3 Implementation details

The previous proposed methodology has been implemented by means of several programs which are external to the base XML information retrieval system being considered.

The first program, the query processor, parses the NEXI query and extracts the different content-only subqueries. These are passed to the XML retrieval system, which runs a retrieval for each subquery and obtains a ranking of relevant elements. These ranked outputs are stored in (text) files, containing the document identifier, the XPath and the RSV of each element.

These files are then processed by another program, the output processor, which in turn calls (according to the structure of the original query) three filters implemented as perl scripts.

The first script removes the results that do not satisfy the structural restrictions for each subquery and computes the aggregated probabilities of the structural units of the same type, using an OR gate in Equation (4.1); this filter is executed once for each subquery. A second script then takes the output files

of the first script and computes the probabilities of the context and the target, using either an AND gate or an OR gate in Equation (4.2); this filter is executed twice, once for the target and once for the context. The third script computes the final probabilities by combining the probabilities of the context and the target by using an AND gate in Equation (4.3).

4.4 Related work

In this section, we would like to review various techniques for answering CAS queries found in the specific literature, mainly publications derived from the INEX workshops. We do not wish to make an exhaustive analysis but merely to show the main types of CAS resolution methods and, for those techniques more similar to our proposal, to present the main differences with them. We should also mention that this brief study starts in 2004, the year when INEX made the decision to use NEXI as the language to specify CAS queries. In INEX 2002 and 2003, although this type of query existed in the official tasks, another specification was used.

We have observed two types of approaches in the specialized bibliography for solving structured queries: those in which the retrieval of XML elements given a CAS query is integrated in the XML retrieval model, and those in which CAS queries are managed as an additional, differentiated layer on top of the XML retrieval model. Let us start with the second group as our approach also belongs to it.

Studying several different works describing methods where CAS queries resolution was made on top of a retrieval model, we can easily appreciate how CAS queries are solved in a similar way: content queries of the about clauses are launched to the retrieval systems. Once the rankings are generated, their elements are filtered and aggregated in some way, according to the structural restrictions. The selection of the final elements is made from those which, while satisfying the target path, are also in the ranking of elements fulfilling the context path. In some cases, the resolution method is also supported by the existence and use of several indexes containing information about different XML elements.

Some examples following the previously mentioned CAS queries resolution method are the following:

In [101], the authors design a general algebra to express NEXI queries as probabilistic events, which can then be evaluated using a Bayesian network model as the underlying XML retrieval model.

The base of the paper [61] is a Logistic Regression and Okapi BM-25 algorithm implementation for XML retrieval (plus a combination technique) under several indexes. CAS queries are solved by analyzing the NEXI expression and deciding which indexes to use in the search as well as restricting elements that do not fit the structural restrictions. Each content query contained in the about clauses is run and the results are combined.

Crouch et al. [19] present an extension of the Vector Space Model (VSM) for XML documents. In this case, each document is seen as a set of subvectors, each corresponding to some specific type of element. The content queries of each about clause are run through the targeted subvector. Later, results are merged and the target elements are returned.

In [42], the content of the leaf elements is indexed and additional information is stored (paths from the roots, etc) in a database. The score of each leaf element is computed with a very simple formula, that takes into account the terms in each about clause, and propagated through the remaining nodes in the XML documents. Elements that satisfy the context and target parts of the NEXI query are selected and their scores added. In this line of relevance propagation, and representing the XML documents as trees, in [84], XML retrieval is seen as a relevance propagation process in the tree starting from leaf nodes. A NEXI query is decomposed into subqueries which are then further decomposed into elementary subqueries, and these are propagated first in the tree. The relevance values obtained are aggregated to obtain scores for the subqueries, and in turn these are aggregated to obtain the score of the original query.

Based on the system presented in [42] as the underlying XML retrieval engine, in [99], a CAS evaluation method is shown whereby the original NEXI query is divided into the context and target subqueries: rankings are obtained for each content query from an about clause; each ranking is filtered to match the path constraints from each about clause; projections of score values to the elements at the end of the paths are computed, and the different rankings within a filter are then combined. At the end of these processes, there are two sets of elements:

those satisfying the context query and those satisfying the target query. The output are only those nodes of the target query with a matching ancestor in the context query.

With a multinomial language model as a retrieval model and three indexes (structure, article level, and element level) as the physical representation, in the approach presented in [91, 93], the NEXI query is decomposed into pairs (path, content query). Each subquery, constraining different parts of a document, is run and a ranking of suitable elements obtained. Finally, the rankings must be mixed in order to obtain a single sorted list of elements fulfilling the target path, with an associated score.

Our approach could be easily classified in this category since it is completely independent of the underlying XML retrieval model. We could say that the overall process of the evaluation of NEXI queries presented in this chapter is similar to those presented in [91, 99]. Broadly speaking, the main differences are the combination technique and the selection of the final elements, which in our case are based on a probabilistic approach based on noisy gates. More precisely, the method considered in [99] uses combinatory logic and min-max normalization to combine the different rankings, and the final elements selected are only those target elements that have a matching ancestor in the set of elements obtained for the context query (whereas in our case all the preselected target elements are returned, although those ones whose ancestors do not match with elements in the context query become penalized). In the method proposed in [91], the mixture of the results of the subqueries also uses different operators (maximum and sum). Moreover, the context subqueries contribute to the final RSV of a target element in the same way that the target subqueries. Finally, different ways of decomposing the CAS query into a set of CO subqueries are also proposed in [91].

Now, let us focus on the CAS queries resolution approach where the mechanism of resolution is totally integrated in the XML retrieval model, i.e., given a query the score function usually incorporates some XML features to deal with CAS queries. Broadly speaking, in the specialized literature, we observe how modifications of the Vector Space Model (VSM) and integration of XML in relational databases are the predominant research lines.

With respect to the first line, the main changes in the VSM are the incorporation of XML features to the classic cosine ranking function, the use of different indexes to store the information provided by the different XML elements and the use of additional tree representations, combining scores obtained with the VSM with tree matching. Examples of this category are the following:

In [7], the authors modify the VSM including new XML specific features: the nesting nature of the XML elements and the fact that each element could be retrieved. The authors design a new ranking function similar to the cosine measure for flat documents. To deal with CAS queries, a factor is included to measure how the query structural constraints are satisfied by the paths of the elements. The paper [67] presents a model based on indexing the different types of elements in separate indexes. Based on the VSM, the query is run on each index and the different rankings are merged. For CAS queries, the content query is run against the article index to locate candidates that fulfill the query constraints. In a second step, each part of the query is submitted in parallel to each one of the remaining indexes. A relevance value is computed only for those valid elements from the first step. A third extension is found in [102]. This model is adapted to represent the document structure and enhanced with a tree representation of the queries and documents. NEXI queries are solved by computing content scores using the VSM and making tree matching operations to satisfy the structural restrictions.

Various examples joining XML retrieval with relational databases may also be found. The definition of new query languages adapted to cope with structural restrictions or the mapping of structural queries to SQL are the most common approaches to solve CAS queries by means of DBMSs. A couple of representatives of these two methodologies are [70] and [90]. In the former, a DBMS is adapted to work with XML documents, defining a query language, based on SQL, called DSQL to work with structured documents. Structural restrictions are very easy to deal with as they are naturally included in the DSQL language itself. In the latter, [90], CAS queries are directly mapped to SQL queries. In addition, a score function is introduced in order to produce a ranking of relevant elements.

Other methods where relational databases are used are [95], where index structures are implemented in a relational database, and query resolution is based on

a combination of computing content and structural scores; and [74], as a case combining a text search engine and a native XML database.

A third main line to resolve CAS queries is based on representing the XML documents by means of trees: [5] shows a method based on structural indexes (represented as trees) and element weight computations which takes into account the query and the context of each element. Element weights are calculated according to the content query from each about clause and those with weights greater than 0 are selected. Elements that do not fulfill the corresponding structural restriction are then disregarded. These two steps are repeated until the whole NEXI query is processed, applying the final target structural restrictions to the remaining elements. Another tree-related approach is that presented in [3] which uses a combination of structural summaries, labeled trees representing the XML hierarchical structure of documents and queries, and the BM25 retrieval model. Broadly speaking, the former is used to select document components satisfying the structural restrictions of the NEXI query, and the latter to assign a relevance score and therefore to rank them.

4.5 Experimental evaluation

Garnata will be our base system in the experiments. As we commented in chapter 2, this is our structured information retrieval system based on probabilistic graphical models, but so far it is only able to process non-structured queries.

The first XML document collection considered is the one used in the INEX 2006, 2007 and 2008 editions of the INEX Workshop. In terms of the queries (and the corresponding relevance judgments) used to test our proposal, we have selected the set of queries developed for INEX in the three editions. We have not, however, employed all the original queries which have relevance judgments used in these editions of INEX, but a subset containing 90 queries (34 from 2006, 36 from 2007 and 20 from 2008). For the sake of completeness, the list of these CAS queries is displayed in tables 1, 2 and 3 in the Appendix 10.2.

We have not used either queries which are formally equivalent to content-only queries (i.e. queries of type //*[about(.,text)]) or the majority of queries that while not strictly equivalent to content-only queries are in fact equivalent

if we consider the narrative[1] of these queries (typical examples of this situation are queries such as //article[about(.,text)]). In general, we have used only those queries where the relevance judgments are coherent with the structural restrictions imposed by the CAS queries. In a few cases, we have modified the original CAS query to reach greater coherence between structural restrictions and the narrative of the query. For example, the query //section[about(.,pyramids of egypt)//image[about(.,pyramids)], looks for image elements contained in section elements, whereas the narrative of this query establishes that section elements containing images are looked for. We have then reformulated the query as //section[about(.,pyramids of egypt) and about(.//(figure|image),pyramids)].

In order to obtain the corresponding CO query for each CAS query, we removed all the structural components of the CAS query and kept only the content words. We then ran the Garnata retrieval system using the CO queries (base-CO) and the augmented system using the CAS queries, and compared the results. The weights of the noisy OR and noisy AND gates were fixed (without previous tuning) to constant values[2], $w_o(U_i) = 1$ and $w_a(U_i) = 0.999$.

The measures of retrieval effectiveness are those used in the *focused* task of the INEX 2007 and 2008 *adhoc* track [55], namely the interpolated precision (iP) at selected recall levels (iP[0.0], iP[0.01], iP[0.05] and iP[0.10]) and the average interpolated precision (AiP), all of them averaged across the 90 queries. In the focused task the system must return a ranked list of the most focused document components and the resulting document parts should not overlap. The criterion used to decide, when we find two overlapping components in the raw output generated by Garnata, which to preserve in the final output, is to keep the component having the greatest relevance value and, in case of tie, we keep the more general component (the one containing a larger amount of text) [26].

The results of our experiments are summarized in table 4.1, which displays the corresponding performance measures for the base-CO and the augmented system, as well as the percentage of improvement achieved by the proposed system and the p-value of the statistical paired-t test used to detect significant differences. If the

[1] The narrative is a brief text associated to each query which represents the most authoritative description of the user's information need, and therefore serves as the main point of reference against which relevance should be assessed.

[2] We thought that very high values, close to 1, would be better than lower ones.

p-value is inferior to either 0.05 or 0.01 (significant or very significant difference) we denote this by using "*" or "**" respectively. In fig. 4.4 we also display the recall-precision curves for the 101 recall levels.

	Augmented CAS	Base-CO	% improvement	p-value
iP[0.00]	0.670683	0.467583	43.44	5.27E-10**
iP[0.01]	0.609759	0.406762	49.91	2.98E-11**
iP[0.05]	0.475207	0.332509	42.92	1.00E-06**
iP[0.10]	0.377938	0.294003	28.55	7.71E-04**
AiP	0.143025	0.123672	15.65	9.18E-02

Table 4.1: Comparison between the augmented CAS and the base-CO systems.

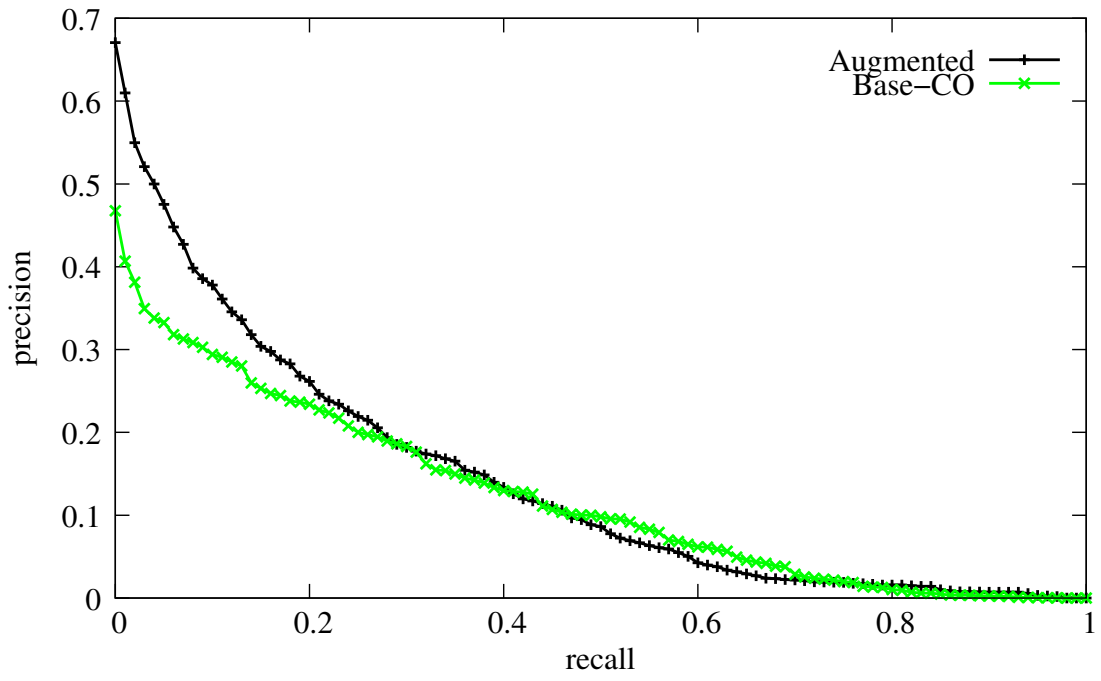


Figure 4.4: Recall-precision curves for the augmented CAS and the base-CO systems.

The results are quite conclusive: the proposed method for managing CAS queries systematically improves the results of the system using CO queries in relation to all the performance measures. Moreover, the improvements achieved

are statistically significant, ranging from a minimum of 15% to a maximum of 50%. We can also observe that the results are much better for lower recall levels. In fact, in fig. 4.4 we can observe that the precision values of the augmented system are always better than those of the base-CO system for all the recall levels until the value 0.4; next, they become worse from 0.41 to 0.76 and again become better from 0.77 to 1.0. This means that adequately processing structured queries seems to concentrate highly relevant document parts in the first positions in the ranking of the results. Consequently, this technique greatly improves the precision and does not significantly deteriorate the recall (which is the initial intuition motivating the use of CAS queries).

In order to see whether the improvement obtained is consistent across the three different sets of queries considered, we display in table 4.2 the performance measures broken down by year. We can observe a behaviour more or less similar across the three years, with the exception of the average interpolated precision, where we get important improvements for the years 2007 and 2008 but a small (not significant) worsening for 2006. This confirms that our method is very effective at lower recall levels but more unstable on the average.

	2006			2007			2008		
	CAS	CO	%	CAS	CO	%	CAS	CO	%
iP[0.00]	0.667	0.432	54.25	0.633	0.437	44.80	0.745	0.582	27.95
iP[0.01]	0.614	0.413	48.78	0.562	0.377	49.05	0.688	0.450	52.95
iP[0.05]	0.513	0.358	43.35	0.444	0.316	40.53	0.466	0.319	46.34
iP[0.10]	0.406	0.323	25.71	0.359	0.281	27.71	0.364	0.268	35.96
AiP	0.156	0.163	-4.50	0.146	0.107	36.01	0.116	0.086	34.74

Table 4.2: Comparison between the augmented CAS and the base-CO systems broken down by year.

In order to study whether our performance improvement is due to simple target restrictions or to the handling of the more complex structural requirements (because the base-CO system can return any element from the collection but our approach can only return elements having the same type as specified by the user), we are going to compare also with another additional baseline (which we shall call base-CAS) which uses the same CO queries as the base-CO system,

but additionally filters away all element that do not match the target element constraint. So, given a CAS query //A[B]//C[D], whereas the base-CO system transforms this query into //*[about(.,content words in B and D)], the base-CAS system transforms it as //A//C[about(.,content words in B and D)]. Table 4.3 and fig. 4.5 display the results of the comparison between our augmented system and the base-CAS system being considered.

	Augmented CAS	Base-CAS	% improvement	p-value
iP[0.00]	0.670683	0.604653	10.92	1.15E-02*
iP[0.01]	0.609759	0.527010	15.70	1.01E-03**
iP[0.05]	0.475207	0.395017	20.30	4.51E-04**
iP[0.10]	0.377938	0.318322	18.73	5.59E-03**
AiP	0.143025	0.116574	22.69	8.99E-04**

Table 4.3: Comparison between the augmented CAS and the base-CAS systems.

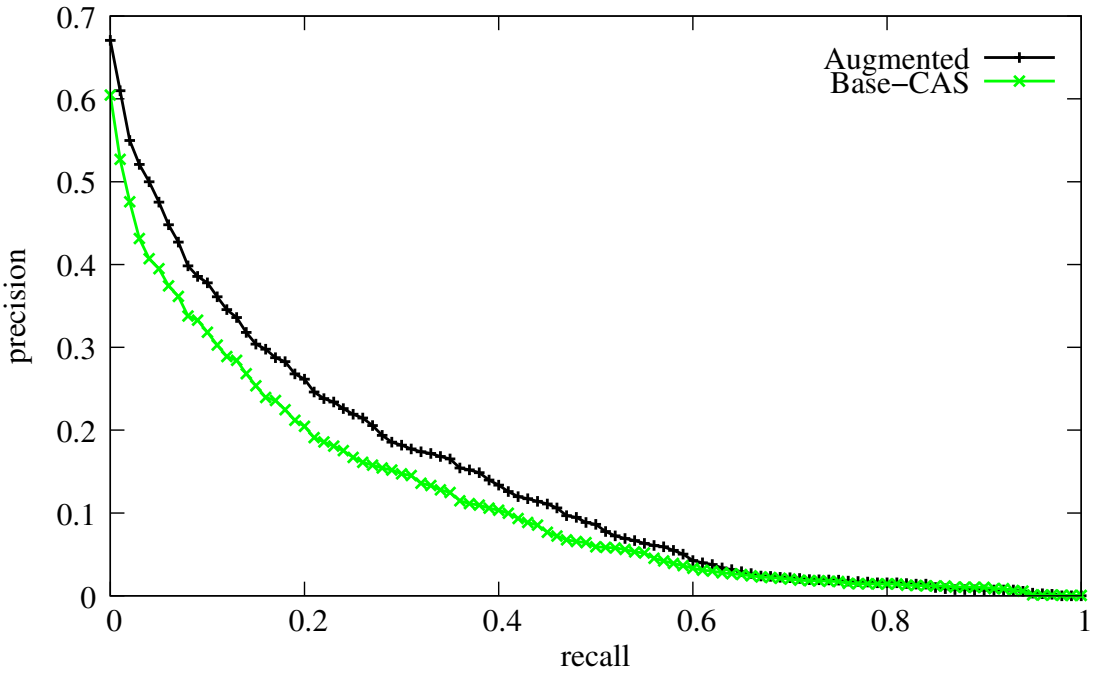


Figure 4.5: Recall-precision curves for the augmented CAS and the base-CAS systems.

We can see that the augmented system still performs significantly better than the base-CAS system with respect to all the performance measures, although in this case the differences are lesser than in the previous case (except with the average interpolated precision). Therefore, we can obtain two conclusions: first, even a simple processing of the CAS queries can lead to improved performance with respect to using only CO queries; second, the proposed method improves further the results of this baseline treatment of CAS queries.

For the sake of completeness, detailed results for individual queries, for the iP[0.01] and AiP measures (which were the "official" measures at INEX [55]) of the augmented CAS, the base-CO and the base-CAS, are displayed in tables 4, 5 and 6 in the Appendix. Table 7 shows the number of queries where our approach is either better, worse or equal than base-CO and base-CAS.

We want also to compare our proposal with another, state-of-the-art method for managing CAS queries. To this end, we have selected and implemented on top of the Garnata system the "full propagation" method proposed by Sigurbjörnsson et al.[1] previously mentioned [91] (that we shall call SKR), which was validated with the INEX *IEEE* Computer Society collection. The results of this comparison are displayed in table 4.4 and fig. 4.6.

	Augmented CAS	SKR	% improvement	p-value
iP[0.00]	0.670683	0.512851	30.78	6.43E-07**
iP[0.01]	0.609759	0.469510	29.87	9.61E-07**
iP[0.05]	0.475207	0.345133	37.69	2.17E-06**
iP[0.10]	0.377938	0.294629	28.28	1.20E-04**
AiP	0.143025	0.120514	18.68	7.25E-03**

Table 4.4: Comparison between the augmented CAS and the SKR systems.

We can observe significant differences in performance between the two methods, favouring again our proposal. In fact the SKR method performs rather poorly in our experimental setting: It behaves worse than the base-CAS except in average interpolated precision, and scarcely ourperforms the base-CO (again with the exception of the average interpolated precision).

[1] Among the different methods studied in their research, this one obtained the best overall results.

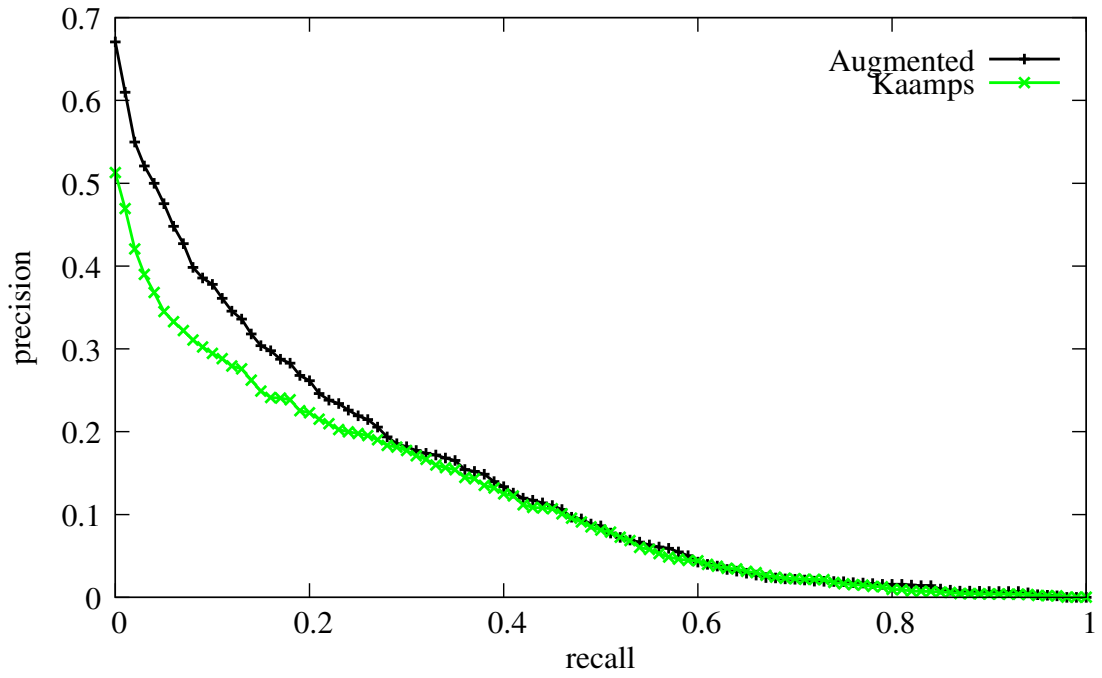


Figure 4.6: Recall-precision curves for the augmented CAS and the SKR systems.

In order to gain insight if there is a direct relationship between base and augmented system performance, we have also carried out experiments with another base system different from Garnata. We have selected the freely available PF/Tijah system[1] [49, 103], part of the MonetDB/XQuery database system. Although PF/Tijah supports several retrieval models, we used the default language model. We have then repeated all the previous experiments but using PF/Tijah instead of Garnata. The results are summarized in table 4.5. We can see that the augmented system always obtains better results than the other three systems, and these differences are still statistically significant for the base-CO and the SKR systems. Therefore, although the behaviour of the proposed method may be somewhat different depending on the base system being used, the trend is to improve the results of the base system.

Another interesting question to test is whether the proposed method for managing structured queries is also able to show consistent improvement using dif-

[1] http://dbappl.cs.utwente.nl/pftijah/

	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	AiP
			Augmented CAS		
	0.568435	0.481378	0.373408	0.303367	0.115764
			Base-CO		
	0.485270	0.388033	0.298495	0.219913	0.067465
%	17.14	24.06	25.10	37.95	71.50
p-value	2.09E-02*	4.22E-03**	1.57E-02*	4.06E-03**	3.46E-04**
			Base-CAS		
	0.530007	0.450899	0.332860	0.249575	0.098816
%	7.25	6.76	12.18	21.55	17.15
p-value	1.64E-01	2.07E-01	1.37E-01	5.21E-02	9.89E-02
			SKR		
	0.415855	0.328691	0.274854	0.233593	0.097079
%	36.69	46.45	35.86	29.87	19.25
p-value	1.56E-05**	4.58E-06**	7.98E-04**	7.30E-03**	4.29E-02*

Table 4.5: Results of the experiments with the *Wikipedia* collection using the PF/Tijah system.

ferent document collections. To this end we have selected the *IEEE* Computer Society collection [57] used in the first years of the INEX initiative. The original collection contained the full-texts, marked up in XML, of 12107 articles of the *IEEE* Computer Society's publications from 12 magazines and 6 transactions, covering the period of 1995-2002, and totalling 494 megabytes in size. In 2005 the collection was extended with new articles from the period 2002-2004, giving a total of 16819 articles and 764 megabytes in size. We have used a subset of 22 CAS queries from INEX 2003[1], 19 from INEX 2004[2] and 14 from INEX 2005[3] (the list of queries is also displayed in the Appendix, in tables 8, 9 and 10).

In the first four editions of INEX (previous to 2006), the relevance judgments were done in a different way, using a two-dimensional scale taking into account exhaustivity and specificity, which were mapped to a single relevance scale by employing several quantization functions. We have used the so-called *strict quantization*, which focuses on components rated as highly exhaustive and

[1] http://inex.is.informatik.uni-duisburg.de:2003/
[2] http://inex.is.informatik.uni-duisburg.de:2004/
[3] http://inex.is.informatik.uni-duisburg.de/2005/

highly specific (and whose exact definition varies slightly across years [56, 64]), as done also in [54, 91, 99].

The measures of retrieval effectiveness are those used in INEX 2005, namely the (system-oriented) non-interpolated mean average effort-precision (MAep) and the (user-oriented) normalized extended cumulated gain (nxCG) at various fixed ranks (nxCG[10], nxCG[25] and nxCG[50]) [56], averaged across the 55 queries. These measures have been computed using the EvalJ evaluation package, more specifically using the XCGEval package[1]. As in the previous experiments with *Wikipedia*, we still consider a focused task where overlap is not permitted (hence the raw results obtained by the IR system are filtered to remove overlapping components). Therefore, we always use the option `overlap=on` when evaluating with EvalJ. Table 4.6 displays the results of these experiments, where the base IR system used has been Garnata. The augmented system once again gets the best results, although the differences with the other systems are somewhat lesser than in the case of the *Wikipedia* collection, specially with respect to the base-CAS system. These differences are not statistically significant in many cases (although in some cases this may be due in part to the less number of queries considered, 55 instead of 90)

Finally, in order to determine the sensitivity of the proposed model to the parameters considered for the AND and OR gates, w_a and w_o, we have carried out another series of experiments varying these parameters. We have used in this case the *Wikipedia* collection and the Garnata IR system. The results are displayed in table 4.7.

The first fact that can be observed in table 4.7 is that the values of the weights w_o and w_a matter, because they have influence on the results. Performance deteriorates systematically as the weight of the AND gate decreases, excluding the weight $w_a = 1.0$ (pure AND gate), which produces disastrous results. Therefore, a very high weight different from 1.0 for the AND gate, like 0.999 gives the best results. The value of the weight for the OR gate, w_o, has also influence on the results but at a lesser extent. The weight $w_o = 1.0$ (pure OR gate) gives almost the best results, together with $w_o = 0.99$. So, our initial selection of weights ($w_o = 1.0$ and $w_a = 0.999$) was, fortunately, a good choice.

[1] https://sourceforge.net/projects/evalj

	nxCG[10]	nxCG[25]	nxCG[50]	MAep
		Augmented CAS		
	0.1681	0.1546	0.1679	0.083731
		Base-CO		
	0.0971	0.1097	0.1220	0.053167
%	73.03	40.94	37.55	57.49
p-value	2.59E-02*	8.91E-02	8.81E-02	1.36E-01
		Base-CAS		
	0.1541	0.1508	0.1676	0.075809
%	9.04	2.50	0.13	10.45
p-value	2.36E-01	4.13E-01	4.94E-01	2.08E-01
		SKR		
	0.1214	0.1276	0.1365	0.060758
%	38.45	21.18	22.99	37.81
p-value	2.41E-03**	2.35E-02*	2.13E-02*	9.37E-03**

Table 4.6: Results of the experiments with the *IEEE* Computer Society collection using Garnata.

In general, our results are considerably better than certain previous results reported in the literature by other researchers [91, 96, 105], who did not find any significant improvement (or even no improvement at all) when managing CAS queries instead of CO queries. Our results are more in agreement with those in [54], who reported improvements in early precision, although we have also obtained good results in average measures. There is no simple explanation for this difference. The naive answer would be to say that some of these previous methods were not able to process structured queries effectively, but probably there are other reasons, such as the different document collections: our results are better for *Wikipedia* than for *IEEE*, which was the collection used to test these methods. This suggests that the effectiveness in managing CAS queries may depend on the type of collection, queries and performance measures being used. Another difference of our experimental setting with respect to previous studies is the type of task considered, focused in our case and thorough in the other cases. Even the base systems considered are also different. Therefore, although the proposed method has demonstrated its effectiveness, we believe that more research efforts should be invested in order to clarify when and how managing

w_o	w_a	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	AiP
1.0	*0.999*	*0.670683*	*0.609759*	*0.475207*	*0.377938*	*0.143025*
1.0	1.0	0.087623	0.082098	0.061717	0.051818	0.024452
1.0	0.99	0.640474	0.571272	0.436222	0.354053	0.131840
1.0	0.90	0.602101	0.538121	0.403327	0.329553	0.123633
1.0	0.75	0.557624	0.498115	0.380840	0.325173	0.119020
0.999	0.999	0.657806	0.590491	0.470164	0.381682	0.139561
0.99	0.999	0.670684	0.609759	0.475160	0.378033	0.143024
0.90	0.999	0.670315	0.607306	0.474859	0.377538	0.142859
0.75	0.999	0.653703	0.586629	0.465401	0.378000	0.137918

Table 4.7: Results of the experiments with the *Wikipedia* collection using Garnata for different weights of the OR/AND gates.

CAS queries is worthwhile.

4.6 Concluding remarks

In this chapter, we have proposed a general methodology for managing structured queries within partially structured IR system able to only deal with content-only queries. Our method can be applied to any probabilistic IR system without the need to modify the system or interact with it in a complex way.

We have demonstrated the effectiveness of our approach by using two specific structured IR systems and two benchmark XML document collections as the test bed. The experimental results confirm that a proper management of the structure present in CAS queries can significantly improve the retrieval capabilities of the system.

Our view of CAS queries in this work has been rather strict in the sense that the target-path constraints present in the query must be upheld for a result to be relevant. If a user asks (for example) for *section* units to be returned, these must be returned (although other aspects of the query are interpreted from the IR perspective, i.e. loosely). There is, however, another less strict, vague interpretation of a CAS query [97], where target-path requirements need not be fulfilled, and the path specifications should therefore be considered hints as to where to look. For future work, we plan to modify our method to deal with this

vague interpretation of CAS queries. Another interesting future research could be to try to take advantage of the flexibility of the noisy-OR/AND used by our approach to improve further the results, by studying how to assign the weights $w(U_i)$ used in these noisy gates in a non-uniform way, depending for example on the type of query being processed or on the type of element being returned.

Chapter 5

Relevance Feedback for XML retrieval

5.1 Introduction

Relevance Feedback (RF), whose basic idea is to do an initial query, get feedback from the user and use this information to create a more adapted query to the user's needs, is one of the objectives of the dissertation, therefore it was introduced in more detail in section 1.2.5 of chapter 1. Then, the focus of this chapter is to show the methodology we have developed to implement our RF framework in Garnata.

It is important to mention that we could classify RF for XML retrieval into three groups according to the types of the original and resulting queries:

- **T1: CO-CO:** The original query and the expanded query are only composed of keywords. New terms are usually added to the original query and term weights (re)computed.

- **T2: CO-CAS:** The original query is only composed of keywords but the expanded query generated by the RF method contains (new) terms, and their corresponding weights, complemented with structural restrictions extracted from the analysis of the assessments.

- **T3: CAS–CAS:** Both queries are composed of terms and structural restrictions. In this case, there are two possibilities:
  - **T3.1:** Not to modify the structural restrictions expressed in the original query but only adding new terms to the underlying CO queries.
  - **T3.2:** In addition to new terms, to modify the existing structural restrictions or including new ones.

RF of T1 and T2 types has been studied in the literature, but there exist few papers dealing with T3 type. In our case, we have already faced the design and evaluation of the types T1 and T3.1, presenting both Content-oriented [25] and Content and Structure-oriented [28] RF techniques on the top of Garnata.

In both approaches we shall study how the original query can be expanded, i.e., the inclusion of new terms (expansion terms) in Q, in order to better capture the user's information needs. Some problems will be considered as how many terms and in which part of the structural restrictions they might be included.

In order to describe the RF techniques, this chapter is organized as follows: Section 5.2 presents some related works about different RF techniques. After that, section 5.3 and 5.4 will show our RF frameworks implemented in Garnata. In order to evaluate the feasibility of our approaches, section 5.5 is devoted to the experimentation. Finally, section 5.6 presents the concluding remarks and points to some future research tasks.

5.2 Related work

RF has been the objective of many researchers as a mean of improving retrieval effectiveness in XML IR. Firstly, we have studied the Rocchio algorithm [78] in chapter 1 in which most of the works undertaken in content RF in structured IR are based on. Ruthven and Lalmas [79] have studied different RF techniques (automatic and interactive techniques), specific interfaces to RF systems and characteristics of searches that can affect the use of RF systems.

We shall describe some works concentrated on query expansion based on the content of elements with known relevance:

- Y. Mass [68] describes a component ranking algorithm for XML retrieval and shows how to apply known RF algorithms from traditional IR on top of it to achieve RF for XML.

- The work of C. Crouch [20] is based on an extension of the vector space model. The major advance achieved is the inclusion of a flexible capability, which allows the system to retrieve at a desired level of granularity (i.e., at the element level).

- In [92], B. Sigurbjörnsson et al. investigate the effectiveness of blind ("pseudo") feedback based on top ranking XML elements.

- Pan et al. [72] apply user feedback to recompute similarities in the ontology used for query evaluation.

On the other hand, several papers have considered structural query expansion: The first work is developed by Mihajlovic et al. [69] to extend their database approach. They assume that knowledge of component relevance provides "implicit structural hints" which may be used to improve performance. Their implementation is based first on "extracting the structural relevance" of the top-ranked elements and then restructuring the query and tuning the system based on RF information. They argue that the document names of the relevant components are used to model structural relevance because these documents are apt to contain similar information. Using the structural information and assessments associated with the relevant elements, the query is rewritten and evaluated.

The second approach, proposed by Schenkel and Theobald [85], consists of extracting classes of features for each relevant element. These classes of features are: the ancestor class, the descendant class and the content class. For example, for a given relevant element section, article and body elements are added to the ancestor class, paragraph and subsection elements are added to the descendant class, and terms of the section element are added to the content class.

Fourati et al. [32] propose an approach where they take the original CAS query and the fragments judged as relevant by the user. Then, they create a representation of the original query and relevant fragments under a matrix form. After some processing and calculations on the obtained matrix and after some

analysis they have been able to identify the most relevant nodes and their relationships that connect them. They are used to modify the structure of the new query but keeping the same content.

Finally, L. Hlaoua et al. [50] propose to add structural constraints to the initial keyword query. Their approach first seeks to identify the generic structure shared by the largest number of relevant elements and then they use this information to incorporate in the expanded query.

5.3 A proposal for content-oriented Relevance Feedback

Our content oriented RF method using Garnata aims at identifying elements, as a result of solving a keyword query, that are relevant and non-relevant, based on the decision of the user, to use some of the terms contained in them to formulate a new expanded query. Once the terms have been extracted, the system computes weights that represent the corresponding probabilities of each selected term given the query. As a consequence of this process, we obtain a new query with the original terms and, perhaps, some new terms, where every term has a different associated probability. This is the main difference with respect to the original keyword queries which only contain relevant terms. For instance, "XML format", whose relevance degree for each term is always 1 (the maximum probability). If we transform the example keyword query into a weighted query with probabilities, it would be "1.0*XML 1.0*format".

Among the terms contained in the elements judged as relevant or non-relevant by the user, we shall focus on the following subsets:

- RQ terms: terms in the original query that only appear in relevant elements.
- NRQ terms: terms in the original query that only appear in non-relevant elements.
- NQ terms: terms in the original query that appear in both relevant and non-relevant elements.

- RT terms: terms which do not appear in the original query and only appear in relevant elements.
- NRT terms: terms which do not appear in the original query and only appear in non-relevant elements.

We can see an example of the terms used in expanded queries in fig. 5.1, where the original query is $Q = \{t_3, t_4, t_6\}$:

- RQ candidates: t_6.
- NRQ candidates: t_3.
- NQ candidates: t_4.
- RT candidates: t_1 and t_5.
- NRT candidates: t_2.

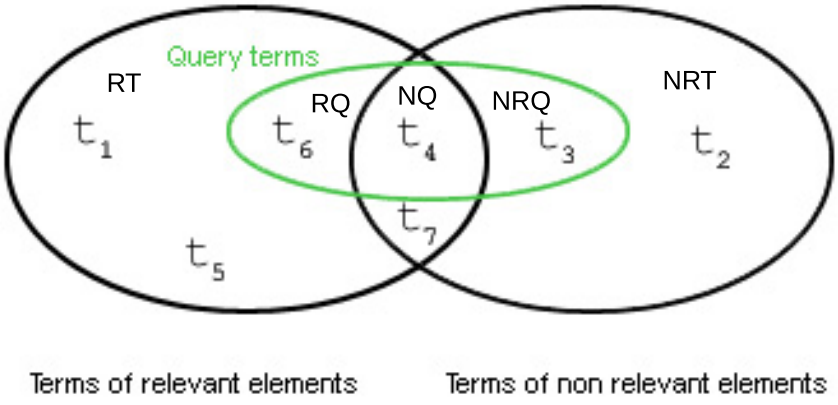


Figure 5.1: Candidate Terms.

5.3.1 Generating the new query

Using the candidates described before, the IR system builds a new content-only query from the original content query that is better adapted to the needs of the user. The new query presents the special feature that it contains the probabilities

of the terms given the query, which determine the importance of each term of the query. To a better understanding of this new expanded query, its general structure is:

$$t_{RT} * p(t_{RT}|Q)\ t_{NRT} * p(t_{NRT}|Q)\ t_{RQ} * p(t_{RQ}|Q)\ t_{NRQ} * p(t_{NRQ}|Q)\ t_{NQ} * p(t_{NQ}|Q).$$

As an example, if the original query was "XML format" and we select the RQ candidate "XML" with probabilitiy 1.0, the NRQ candidate "format" with probability 0.5, the RT candidate "metadata" with probability 0.8 and the NRT candidate "Java" with probabilitiy 0.0, the expanded query would be:

0.8*metadata 0.0*Java 1.0*XML 0.5*format.

In the last step, Garnata runs this expanded query, so we have modified the implementation of Garnata for this new type of queries, because the inclusion of (different) probabilities associated to terms in the query was not accepted by the original version of Garnata.

5.3.2 Computation of probabilities of relevance for original and expansion terms

In this section, we are going to describe how the probabilities of term relevance can be computed, given the query, for those terms belonging to the query and for those candidate for query expansion. These are the different alternatives:

- *RQ terms*: The probability for this type of candidates is 1.0 because these terms are doing well their job, so they are very significant:

$$p(t_{RQ}|Q) = 1.$$

- *NRQ terms*: This type of candidates must appear in the new query because they are part of the original content query, but they are not performing well. Consequently, they should be penalized by decreasing the belief supporting their relevance. So, we propose that this penalization was a function of the number of non-relevant elements in which they are contained:

$$p(t_{NRQ}|Q) = \frac{1}{n_{t\bar{r}}+1},$$

where $n_{t\bar{r}}$ denotes the number of non-relevant elements that contain the candidate term t.

- *NQ terms*: In spite of these candidates appear in non-relevant elements, their probability is still fixed to 1.0 because they are original query terms and are also contained in relevant elements:

$$p(t_{NQ}|Q) = 1.$$

- *RT terms*: The associated probability given the query Q for this type of terms can be computed in the following two ways:
  - The importance of a term depends on the number of relevant elements in which it appears:

$$p(t_{RT}|Q) = \frac{n_{tr}}{n_r}. \tag{5.1}$$

    Here, for a candidate term t, n_{tr} denotes the number of relevant elements that contain t, and n_r denotes the number of relevant elements judged.
  - This is an extension of the previous one so the functionality is the same, however we want to penalize the terms that are very common in the document collection:

$$p(t_{RT}|Q) = \frac{n_{tr}}{n_r} * \frac{idf_t}{maxidf_{RT}}, \tag{5.2}$$

    where idf_t is the *idf* of the term t and $maxidf_{RT}$ is the maximum *idf* of all the *RT* candidates.

- *NRT terms*: The probability for this type of candidate terms is 0 because we want to penalize the terms that only appear in non-relevant elements:

$$p(t_{NRT}|Q) = 0.$$

5.3.3 Inference and decision making with expanded queries

The inference and decision making process changes when we are using expanded queries because all the terms do not have the same importance in this kind of queries as in the original ones (the importance of all the query terms is the same, so the probability of every term is 1). Then, we have to take into account this feature in the computation of the expected utility of the structural units. If a unit contains some query terms, it does not have to mean that it is very relevant due to it depends on the importance of each term too. For instance, we can have a unit containing several query terms with very low probabilities, i.e., low importance. Then, this unit is not so relevant.

We have to remember that the main functionality of Garnata is, given a query, to compute the expected utility of retrieving each structural unit, and then to give a ranking of those units in decreasing order of expected utility. As we have seen in chapters 2 and 3, the CID model achieves this task, although there is a new challenge we have to face because the queries that Garnata processes, contain only terms. So, this search engine needs several changes to process augmented queries which contain terms and their corresponding relevance probabilities, i.e. their importance.

Therefore, we have to include the importance of the terms in the computation of the posterior probabilities of the structural units in the CID model, but we only have to modify the probability of the basic units because they are the only units containing terms which have importance values. So, the probability of the complex units does not suffer any modification.

As we can see in the formula 2.9, we have to change the second part of it corresponding to the sum of the weights of the query terms since we must weight it with the importance of these terms. To include the importance values in the formula, we multiply the weight associated to each term T belonging to the basic unit B with the importance of the term T in the query q. Then, the new formula

is:

$$\forall B \in \mathcal{U}_b, \qquad p(b^+|q) = \sum_{T \in Pa(B) \setminus Q} w(T,B)\, p(t^+) + \sum_{T \in Pa(B) \cap R(q)} w(T,B) p(t^+|q)\,, \tag{5.3}$$

where $p(t^+|q)$ is the probability of the term given the expanded query.

On the other hand, in Garnata the utility value V_u of each structural unit U is made of a component which depends on the involved unit, a second component which depends only on the type of tag associated to that unit, and a third one independent on the specific unit (these three components are multiplied in order to form the utility value as presented in formula 3.2). In concrete, we are interested in the part depending on the involved unit which is defined in formula 2.6 as the sum of the inverted document frequencies of those terms contained in U that also belong to the query Q, normalized by the sum of the idfs of the terms contained in the query because this part is affected by the importance of the terms.

We can not use the original formula for expanded queries because it considers all the query terms have the same importance. So, they can not be instantiated to values from 0 to 1. Then, we have to reformulate this equation taking into account the probabilities of the terms given the query like in the computation of the probabilities. Thus, we have weighted the sum of both numerator and denominator with the importance of the query terms, so the new formula is:

$$nidf_Q(U) = \frac{\sum_{T \in An(U) \cap Q} idf(T) * p(t^+|q)}{\sum_{T \in Q} idf(T) * p(t^+|q)}, \tag{5.4}$$

where $p(t^+|q)$ is the probability of the term given the expanded query.

5.4 A proposal for content-oriented Relevance Feedback of Content and Structure queries

In this section we are going to present our CAS-CAS RF approach. In general terms, our proposal focuses on creating a new expanded CAS query keeping the same structural restrictions as the original one but expanding the different content

subqueries of the context and target parts (the content queries included in each about clause of the NEXI query). This is one of the main diferences between our proposal and those in the literature (see section 5.2).

5.4.1 Relevance Feedback and Content and Structure queries

Once the whole CAS query is submitted to the IR system, for instance (in natural language) "sections about 'Asia'" in articles dealing with "global warming", and it has returned the set of relevant sections, the user inspects the ranking and makes the corresponding relevance assessments. Then, the system analyzes the judged sections and extracts new terms which will be used to expand the query.

In this section we propose to expand both parts of the CAS query, i.e., those parts related to the target and the context structural restrictions, respectively. As a consequence a new CAS query will be created by adding some terms in the about clauses. Following with our example, the new query could be "sections about 'Asia China' in articles about 'global warming heating'". In this case, China and heating have been used to expand the two subqueries, the first one related to the target and the second one related to the context.

In order to do that, the system must be able to refine the existing simple subqueries independently on their location in the CAS query. In the expansion of the target subqueries, there is no problem because we know the relevance values of related elements, but this situation does not hold for the rest of the structural restrictions, i.e., context restrictions. This is due to the units that might be considered to extract the terms for these subqueries are usually different to the type of the target unit. Therefore, the user did not assess relevance judgments on these units.

The problem is how to find those appropriate structural elements from where to select the expansion terms. In order to tackle this problem, in this section we are going to consider the following assumptions:

- **Relevant Assumption**: If a user judges a target element as relevant, then all the context elements used to generate the final RSV are considered also relevant.

$J(E)$	$J(E^t)$	$J(E^c)$	
r	r	r	
		Hard	Soft
nr	nr	nr	-

Table 5.1: Relevance Assumptions.

- **Non-Relevant Assumption (Hard)**: If a user judges a target element as non-relevant, then all the context elements used to generate the final RSV are considered also non-relevant.

This last assumption can be considered very restrictive, so we propose also to consider a light version:

- **Non-Relevant Assumption (Soft)**: If a user judges a target element as non-relevant, then we can not infer any relevance value about the context units used.

By means of these assumptions we can infer the relevance value for other different units to the ones the user inspects. Table 5.1 summarizes the different alternatives, where $J(E)$ represents the user opinion of the element E and the columns named with E^t and E^c refers to the inferred judgments of target and content elements, respectively. Therefore, to perform term expansion in CAS–CAS RF we have to keep track of the units used to compute the relevance degree for each element presented to the user. These units will be used to select the most useful terms to expand the content subqueries, and to compute their weights representing their importance.

In order to illustrate this process we shall consider the following example where the target is any section included in an article:

Find those sections having a paragraph about NEXI in a article dealing with *information retrieval*.

```
//article[about(.,information retrieval)//section[about(//p,
                                NEXI)]
```

Left hand side of fig. 5.2 shows three results presented to the user and the assessed relevance judgments. According to the method described in chapter 4 to solve CAS queries, these results have to be obtained by the combination of the different units retrieved after submitting to the system the set of simple subqueries. Thus, if we denote by $R(Q,k)$ the k^{th} result for query Q, we have that

$$R(Q,k) = \otimes_{i=1}^{m} R_k(Q_i),$$

$Q_1, ..., Q_m$ being the set of simple subqueries in Q and $R_k(Q_i)$ each one of the intermediate results. In our example, we have two simple CAS subqueries $Q_1 =$ `//article[about(., information retrieval)]` and
$Q_2 =$ `//article//section[about(//p, NEXI)]`. The first one returns a set of articles and the second one returns a set of sections containing a paragraph.

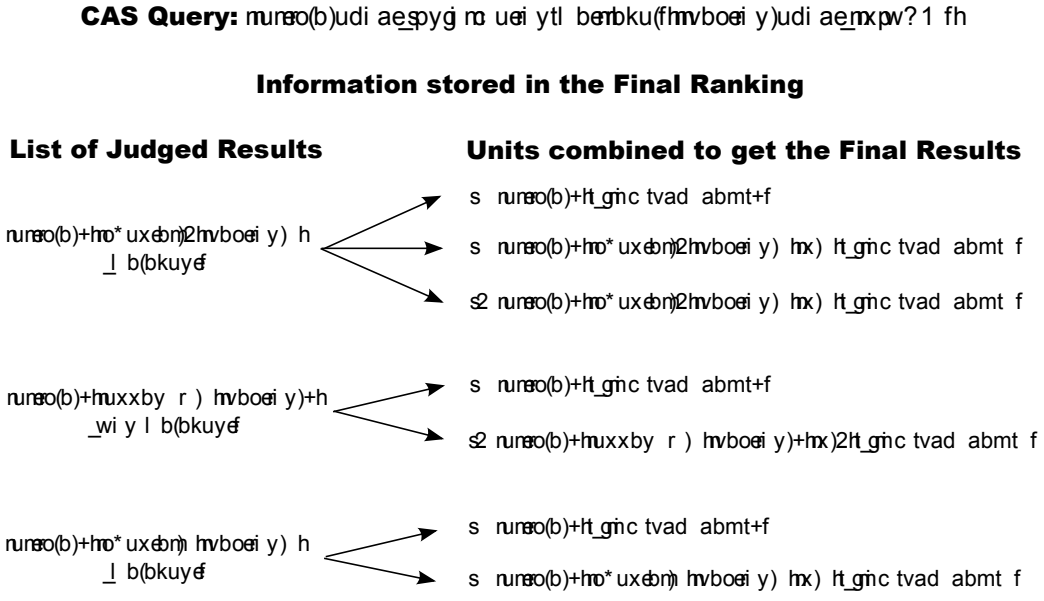


Figure 5.2: Assessed elements with their corresponding set of generators.

Let us focus on $R_k(Q_i)$ since they will have a special role in our CAS-CAS RF approach. In this case, it is also possible that there exist several units, which are results of the subquery, playing a role in the creation of the result $R_k(Q_i)$, and as a consequence candidate units from where we might extract the expansion terms.

We named these units the *set of generators*, $\mathscr{G}_k$, i.e. relevant units in a subquery satisfying at the same time the structural restrictions.

Let us follow with our example by focusing on $R_1(Q_2)$, i.e. the results of Q_2 involved in the first judged result of Q. In this case it is possible to have several paragraphs including the term NEXI in a same section. Particularly, looking at right hand side of fig. 5.2 (the format is: *RSV * unit XPath* and the subquery that they satisfy) we found that the paragraphs `p[2]` and `p[4]` in `/article[1]/chapter[3]/section[4]` satisfy this property, and they are the set of generators for $R_1(Q_2)$.

In order to speed up the query expansion process, for each set of generators only one unit (the one having the highest RSV) is considered as a candidate to extract the expansion terms. Thus,

$$R_k(Q_i) = arg_{u \in \mathscr{G}_k} maxRSV(u, Q_i).$$

Then, the union of these for all the n judged results are the units satisfying the structural restrictions associated to each subquery, named the *set of subquery expansion units (SEU)*

$$SEU(Q_i) = \bigcup_{r=1}^{n} R(Q_i, r).$$

In our example we have the following set of subquery expansion units:

- $SEU(Q_1)$: `/article[1]`
- $SEU(Q_2)$:
  - `/article[1]/chapter[3]/section[4]/p[2]`
  - `/article[1]/appendix[5]/section[1]/p[3]`
  - `/article[1]/chapter[2]/section[2]/p[7]`

5.4.2 Selecting the expansion terms

Each subquery Q_i will be expanded by selecting the terms from $SEU(Q_i)$. This is one of the differences from our previous approach for CO-CO RF, where the terms have been selected from those judged units in the final ranking. In this process there are two different, but related, problems: Which the expansion terms are and what the weights associated to each one are.

Before dealing with these problems, it is important to mention that we do not modify the original query terms, i.e., we do not perform query term reweighting, do not add non-relevant terms with low weights to the expanded query and use the formula 5.5 to compute the weights of the relevant expanded terms because we follow the configuration of the best alternative of our CO–CO approach, as we shall see in the experiments of section 5.5.3.

In order to deal with the first problem in this section we are going to assume that only those terms belonging to relevant elements in $SEU(Q_i)$ but not belonging to any non-relevant elements in this set can be considered as candidates for the expansion of the subquery Q_i as we have commented previously. In other words, a term in a non-relevant element can not be an expansion term. We are assuming that the content of these elements, which have a great literal similarity with the query, is not related with the query intent.

Therefore, by taking up again the assumptions in section 5.4.1, summarized in table 5.1, the candidate terms for a subquery Q_i will be determined. Nevertheless, depending on the type of the non-relevant assumption (soft or hard) considered, the set of candidate terms might differ. In our example, focusing on the subquery Q_2, a term t_i that belongs to both, `/article[1]/chapter[3]/section[4]/p[2]` and `/article[1]/appendix[5]/section[1]/p[3]` is a valid candidate if we use the soft non-relevant assumption whereas this term can not be a valid candidate when considering the hard version.

Once we know the candidate terms, the second step will be to select the best ones. In order to deal with this problem, a weight measuring the importance of each term will be used. Then, those k candidate terms with the highest importance weights expands the corresponding subquery. Particularly, for a given subquery Q_i, it is assumed that the importance of a candidate term t depends

on the number of relevant elements in $SEU(Q_i)$ in which it appears. Then it is computed according to the following expression which is the used one in the configuration of the best alternative of the CO–CO approach (see formula 5.1):

$$w(t) = \frac{n_{tr}}{n_r}, \tag{5.5}$$

n_{tr} being the number of relevant elements of the set $SEU(Q_i)$ that contain t, and n_r denotes the total number of relevant elements of that set.

Finally, the expanded CAS query can be formed by adding the k top-weighted terms to the original query. In this new query the original query terms are weighted with 1.0 and the expanded terms with their corresponding importance weights. For example, selecting two expansion terms, the expanded query might be:

```
//article[about(.,1.0*information 1.0*retrieval 0.5*search
0.4*document)//section[about(//p, 1.0*NEXI 0.8*XML 0.7*structure)]
```

5.5 Experimental evaluation

5.5.1 Data set and evaluation measures

We have performed several experiments with the collections, CO and CAS queries at INEX 2006 and INEX 2007 in order to validate our proposals.

In terms of the queries (and the corresponding relevance assessments) used in our experiments, we have selected the set of queries developed for INEX 2006 (114 CO queries and 34 CAS queries) and INEX 2007 (103 CO queries and 36 CAS queries). Table 5.2 shows some statistics of the CAS queries. Second column presents the numbers of queries which do not have context part. Third and fourth columns show the mean number of subqueries for the target and context component, and fifth and sixth present the mean lengths of the queries.

With respect to the evaluation, we have considered the *focused task* of the main INEX *adhoc* track. The objective is to retrieve the most relevant parts of the documents, without overlapping (for example, it is impossible to retrieve a section and a paragraph of this section simultaneously).

Year	x	$n(Q^t)$	$n(Q^c)$	$\mathrm{l}(Q^t)$	$\mathrm{l}(Q^c)$
2006	5	1.24	1.03	2.32	2.12
2007	19	1.39	1.12	1.73	1.94

Table 5.2: Query statistics.

The measures of retrieval effectiveness are those used in the focused task of the INEX 2007 *adhoc* track, namely the interpolated precision (iP) at selected recall levels (iP[0.0], iP[0.01], iP[0.05] and iP[0.10]) and the average interpolated precision (AiP), all of them averaged across the CO and CAS queries.

5.5.2 Evaluation methodology

The objective is to compare the results returned by the search engine for the original and expanded queries, which are generated after knowing the relevance of the first m structural units of the result list (this relevance information is obtained from the relevance assessments of INEX). So, we can test if the expansion of CO and CAS queries with our approaches is interesting.

This comparison is not an easy task because the results from the expanded query could apparently be very positive due to the fact that the elements judged as relevant will probably appear in the first positions of the ranking of the expanded query (this fact is called *ranking effect*). The use of this data for the *training* in the evaluation has an overfitting effect which artificially improves the results. For this reason, there are different techniques to evaluate RF methodologies like *residual collection* or *freezing* [15].

In our case, we shall use the residual collection method, but it has been adapted for structured documents (because we must take into account that if one unit has been judged as relevant, then we have relevance information about other units which could appear in the result list being ancestors or descendants of it) and the *focused task*. Therefore, the method works as follows:

- **Original query**: We consider the original ranking of the system (with all the overlapping units removed). Afterwards, the first m judged elements are removed too. This ranking will be considered the baseline.

- **Expanded query**: The system obtains the ranking for the expanded query (without removing the overlapping units). In addition, the m judged elements are added at the beginning of the ranking. Then, a filter is used to remove all the overlapping units and finally, the previous m judged elements are removed.

Then, these two retrieval lists could be compared in order to measure the impact of RF, computing a percentage of change in the performance measures.

5.5.3 Experimental results for the Content-oriented approach

Different experiments using the set of queries from INEX 2007 (103 CO queries) have been performed in order to determine the impact of query term re-weighting and query expansion isolately and also combining both approaches. These experiments are:

- Exp. 1: This experiment is a query term re-weighting, i.e., the expanded query is only composed of RQ, NRQ and NQ terms.
- Exp. 2: Original query (query terms with probabilities equal to 1) and the top 10 RT candidates (those with higher probability of relevance) using eq. 5.1.
- Exp. 3: Original query (query terms with probabilities equal to 1) and the top 10 RT candidates using eq. 5.2.
- Exp. 4: Original query (query terms with probabilities equal to 1) and those top 10 NRT candidates which appear most frequently in non-relevant units.
- Exp. 5: Query term re-weighting (RQ, NRQ, NQ) and expanded query using the top 10 RT terms using eq. 5.1.
- Exp. 6: Query term re-weighting (RQ, NRQ, NQ) and expanded query using the top 10 RT terms using eq. 5.2.

- Exp. 7: Query term re-weighting (RQ, NRQ, NQ) and expanded query using the top 10 RT terms using eq. 5.1 and the top 10 NRT elements.
- Exp. 8: Query term re-weighting (RQ, NRQ, NQ) and expanded query using the top 10 RT terms using eq. 5.2 and the top 10 NRT elements.

The results of these experiments, judging the first 10 results, are summarized in tables 5.3, 5.4 and 5.5. They display the corresponding effectiveness measures for the baseline (Base-CO) and for each one of the above experimental settings. The percentage of improvement achieved by these experiments is also indicated.

	Base-CO	Exp. 1	% imp.	Exp. 2	% imp.	Exp. 3	% imp.
iP[0.00]	0.338646	0.352185	3.99	0.325602	-3.85	0.339003	0.11
iP[0.01]	0.296273	0.301896	1.90	0.312909	5.61	0.317543	7.18
iP[0.05]	0.210609	0.211884	0.61	0.268876	27.67	0.244061	15.88
iP[0.10]	0.175126	0.176834	0.98	0.243192	38.87	0.199332	13.82
MAiP	0.058443	0.058556	0.19	0.085947	47.06	0.080139	37.12

Table 5.3: Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement)

	Base-CO	Exp. 4	% imp.	Exp. 5	% imp.	Exp. 6	% imp.
iP[0.00]	0.338646	0.342393	1.10	0.342393	1.10	0.328314	-3.05
iP[0.01]	0.296273	0.296757	0.16	0.316852	6.95	0.311541	5.15
iP[0.05]	0.210609	0.210940	0.16	0.245233	16.44	0.267845	27.18
iP[0.10]	0.175126	0.174750	-0.21	0.200200	14.32	0.242914	38.71
MAiP	0.058443	0.058409	-0.06	0.080040	36.95	0.085852	46.90

Table 5.4: Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement)

The results of our experiments are quite conclusive: Our RF module obtains, in most cases, better results than the baseline system. Nevertheless, the best results have been obtained expanding the query using RT terms. With respect to query term re-weighting, Exp. 1 shows the best results when it is applied isolately. In this case, we can see that using different probabilities for the term queries does not affect too much to the results. Moreover, when query term re-weighting is

	Base-CO	Exp. 7	% imp.	Exp. 8	% imp.
iP[0.00]	0.338646	0.326174	-3.68	0.348205	2.82
iP[0.01]	0.296273	0.309401	4.43	0.322664	8.90
iP[0.05]	0.210609	0.261159	24.00	0.251045	19.20
iP[0.10]	0.175126	0.241597	37.96	0.197924	13.02
MAiP	0.058443	0.085368	46.07	0.080163	37.16

Table 5.5: Comparison between the different Relevance Feedback experiments and the base-CO systems. (% imp. = % improvement)

used in combination with term expansion (Exp. 5 to 8) the performance decreases. We believe that this performance is due to the short length of the queries (a typical query contains 3 or 4 terms) and the fact that these terms could be selected carefully by the user. With respect to query expansion, using RT terms (Exp. 2 and 3) is the best solution because the improvements achieved are highly significant, ranging from a minimum of 37% to a maximum of 47%. The best results have been obtained using eq. 5.1, i.e. without considering the importance of the term in the collection.

Finally, the expansion of the query using NRT terms (Exp. 4 isolately and Exp. 7 and 8) seems that they do not affect the effectiveness of the system. This performance is due to the low prior probabilities associated to the terms, $P(t)$. Thus, we have that for a given term, t, in NRT the $P(t|q)$ is quite similar to $P(t)$. So, there is not going to be a big difference if the system propagates the prior probability (in the baseline) or zero (in RF approach) for NRT terms.

After seeing these results, we want to emphasize the importance of the number of terms used in the expansion of the queries. In fact, we decided to continue studying about this matter and its influence in the quality of the results in Garnata. Therefore, we tried to get some useful conclusions.

Following the configuration of the best experiment of our first approach, Exp. 2, different experiments have been performed changing both the number m of judged structural units and the number k of terms added to the query. In concrete, we have experimented with the values $m = 5, 10, 20$ and $k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$, in total 30 experiments (the input of them has been the 217 CO queries of INEX 2006 and INEX 2007). Finally, there is a experiment selecting for each query the best number of expanded terms (best), this

experiment will be described in more detail. The results of these experiments are shown in tables 5.6 and 5.7.

	$m = 5$				$m = 10$			
k	iP[0.01]	iP[0.05]	iP[0.10]	AiP	iP[0.01]	iP[0.05]	iP[0.10]	AiP
without RF	0.384	0.309	0.254	0.096	0.334	0.245	0.197	0.069
1	0.403	0.345	0.298	0.112	0.351	0.279	0.223	0.079
2	0.379	0.337	0.290	0.112	0.343	0.271	0.228	0.084
3	0.393	0.340	0.297	0.118	0.370	0.295	0.256	0.092
4	0.374	0.327	0.288	0.113	0.365	0.307	0.262	0.095
5	0.382	0.339	0.301	0.119	0.358	0.294	0.256	0.095
6	0.376	0.330	0.294	0.116	0.351	0.298	0.249	0.096
7	0.367	0.319	0.285	0.116	0.350	0.287	0.240	0.094
8	0.353	0.309	0.281	0.114	0.337	0.282	0.250	0.094
9	0.333	0.298	0.271	0.112	0.333	0.278	0.248	0.095
10	0.318	0.285	0.267	0.110	0.316	0.268	0.240	0.093
best	0.501	0.444	0.393	0.152	0.489	0.393	0.339	0.124

Table 5.6: Results of the experiments in the following alternatives: Without Relevance Feedback, using m judged units and k expanded terms, and the best number of expanded terms for each query, with $m = 5, 10$.

	$m = 20$			
k	iP[0.01]	iP[0.05]	iP[0.10]	AiP
without RF	0.261	0.180	0.122	0.046
1	0.323	0.234	0.174	0.062
2	0.306	0.235	0.190	0.066
3	0.304	0.237	0.195	0.070
4	0.312	0.239	0.198	0.073
5	0.308	0.246	0.203	0.075
6	0.314	0.256	0.209	0.075
7	0.307	0.250	0.209	0.074
8	0.296	0.241	0.202	0.074
9	0.295	0.239	0.204	0.074
10	0.293	0.247	0.207	0.076
best	0.439	0.351	0.278	0.098

Table 5.7: Results of the experiments in the following alternatives: Without Relevance Feedback, using m judged units and k expanded terms, and the best number of expanded terms for each query, with $m = 20$.

It is more conclusive to observe the tables 5.8 and 5.9, where the percentages of improvement from the results of RF compared to the results without RF have been computed. Firstly, we can emphasize RF improves almost all the results (except when $m = 5$ and low recall values), although these results are very variable, from changes insignificant to changes very remarkable. Then, it seems that RF is, in general, a useful technique in structured IR too.

	$m = 5$				$m = 10$			
k	%iP[0.01]	%iP[0.05]	%iP[0.10]	%AiP	%iP[0.01]	%iP[0.05]	%iP[0.10]	%AiP
1	4.92	11.70	16.95	16.13	5.13	13.83	13.12	14.27
2	-1.23	9.29	14.05	16.41	2.92	10.91	15.47	22.64
3	2.34	10.22	16.56	22.39	10.86	20.52	29.77	33.16
4	-2.61	5.95	13.11	18.01	9.54	25.44	32.61	37.48
5	-0.51	9.70	18.18	23.94	7.18	19.98	29.84	37.51
6	-2.14	6.83	15.63	21.07	5.22	21.67	26.11	38.97
7	-4.38	3.24	12.11	20.93	4.74	17.16	21.86	37.13
8	-7.93	-0.03	10.44	18.51	0.80	15.26	26.88	36.86
9	-13.37	-3.37	6.62	16.11	-0.27	13.50	25.74	38.16
10	-17.26	-7.53	4.84	14.35	-5.34	9.43	21.92	35.01
best	30.65	43.77	54.29	58.42	46.68	60.57	71.91	80.42

Table 5.8: Improvement percentages obtained comparing the results with Relevance Feedback to the results without Relevance Feedback, using $m = 5, 10$.

	$m = 20$			
k	%iP[0.01]	%iP[0.05]	%iP[0.10]	%AiP
1	23.90	29.86	43.11	35.66
2	17.19	30.54	56.04	44.07
3	16.59	31.66	60.43	53.26
4	19.55	33.12	62.94	60.62
5	18.09	37.04	66.63	64.06
6	20.50	42.41	71.70	64.98
7	17.67	38.90	72.05	63.62
8	13.42	33.98	65.61	62.16
9	13.19	32.91	67.74	63.00
10	12.40	37.18	69.85	67.59
best	68.38	94.99	128.09	114.84

Table 5.9: Improvement percentages obtained comparing the results with Relevance Feedback to the results without Relevance Feedback, using $m = 20$.

As we see in fig. 5.3 for the results of tables 5.8 and 5.9, different trends can be appreciated more clearly. Therefore, we obtain the following interesting conclusions:

- Concerning to the results of RF, they improve progressively as the number of judged structural units increases (5, 10 or 20). Our intuition brings us closer to this conclusion because more elements have been judged by the user, more information we have about the nature and the context of the relevant units of a query. Consequently, the RF method works better.

- Results of RF are systematically better as the recall level of the interpolated precision (iP) increases (0.01, 0.05 and 0.10) and considering the

average values (AiP). It seems to indicate RF is a specially good technique to improve recall, although it also improves (in a lower degree) the initial precision measures which are focused on the first obtained results.

- With respect to the best number of expanded terms which generates the best results, it considerably depends on the number of judged units, and on the measure of retrieval effectiveness to a lesser extent. When the number of judged units is low ($m = 5$), it is not useful to expand with a lot of terms, getting the best results with few expanded terms. When the number of judged units is intermediate ($m = 10$), the optimum number of expanded terms increases, ranging from 3 to 10. Lastly, it continues increasing when the number of judged units is higher ($m = 20$), even taking the maximum value of expanded terms, 10. Thus, there is a positive correlation between the number of judged units and the best number of expanded terms for every experiment. Besides, there exists some influence with the considered measure of retrieval effectiveness. Hence, when it corresponds to low recall levels (0.01), the trend is to use less expanded terms than in the case of measures with intermediate or high recall levels (0.10).

It is important to mention that all these comments correspond to experiments where the number of expanded terms is the same for all the queries. In the previous tables and fig. 5.3, the results obtained when we have selected for each query the best number of expanded terms (best) are shown too. These last results represent an upperbound of the improvements we would be able to get a procedure to select the optimum number of expanded terms could be created. This procedure would be based, among others, on the features of the queries.

As we can see in the last row of tables 5.8, 5.9 and fig. 5.3, the improvements are quite good in all the cases, where the improvement percentages respect to not to use RF range from a minimum of 30% to a maximum of 128%. Then, these improvements are better than the best results with a fixed number of expanded terms. In these cases, the percentages are two, three, even six times higher than the previous ones. For instance, in the fig. 5.4 we can distinguish, when $m = 10$ and for all the queries, the differences among the AiP values, when we consider the difference with respect to the baseline (without RF) for both the experiment

with 6 expanded terms or the experiment with the best number of expanded terms (best). In the fig. 5.5, we show the same results when $m = 5$ and iP[0.01].

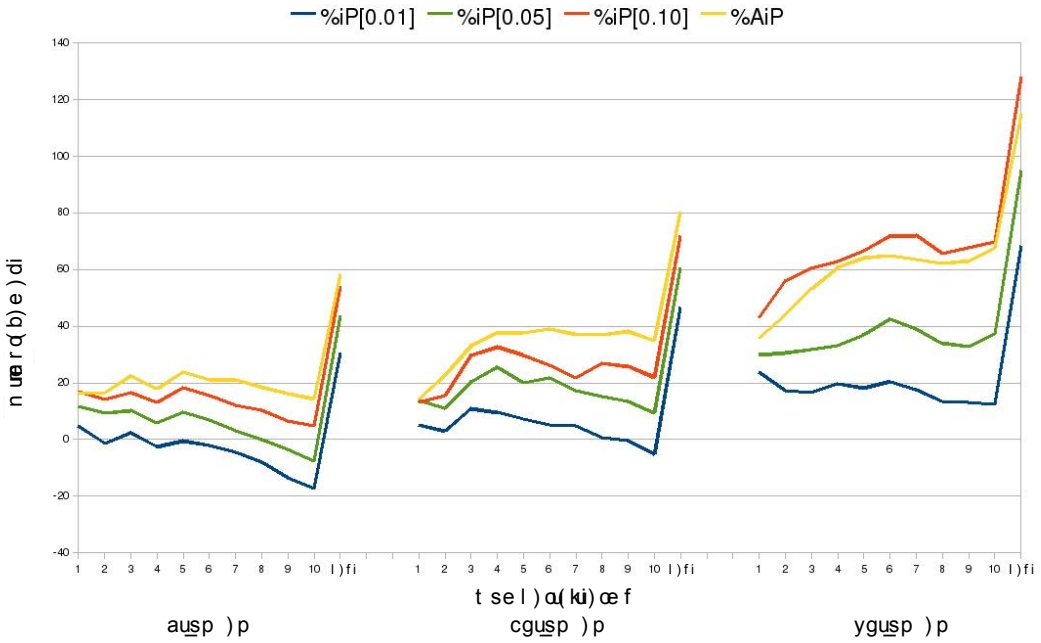


Figure 5.3: Improvement percentages using Relevance Feedback.

It is appreciable it keeps the same behaviour as in the case of a fixed number of terms: Greater is the number of judged units, better are the results. The same happens with the recall level of the performance measures (although in relative terms, the performance improvements, with respect to the obtained ones using a fixed number of terms, are greater conforming the recall level decreases).

In fig. 5.4, but maybe, more clearly in fig. 5.6, there exist a lot of queries with negative results, even if the number of expanded terms is the best case. So, in these cases the use of RF generates worse results. For example, in fig. 5.6 we can distinguish 45 (21%) queries with negative results and 22 (10%) queries where the results are the same in both cases from the 217 queries. In table 5.10,

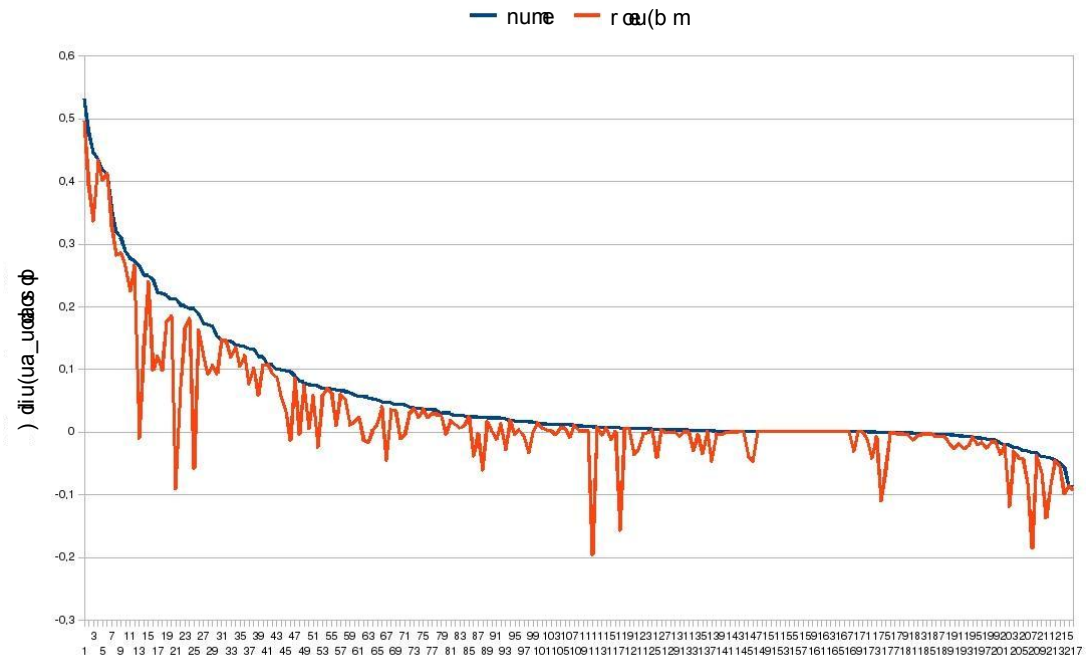


Figure 5.4: Differences in AiP, respect to non-Relevance Feedback case, when we use the best number of terms and the best fixed number of terms (6), for the case $m = 10$.

we indicate the number of queries obtaining worse or equal results for the rest of measures.

In conclusion, RF generates very good results, in general terms, (they are better if the optimum number of expanded terms, we must use for every query, could be determined), although there are several situations where RF is not useful.

5.5.4 Experimental results for the Content and Structured-oriented approach

The first experiment that we have designed and run under this evaluation framework, is based on the expansion of the subqueries using different number of terms and the expansion in different parts of the original query. With the first parameter, we would like to know the impact in the query performance when the

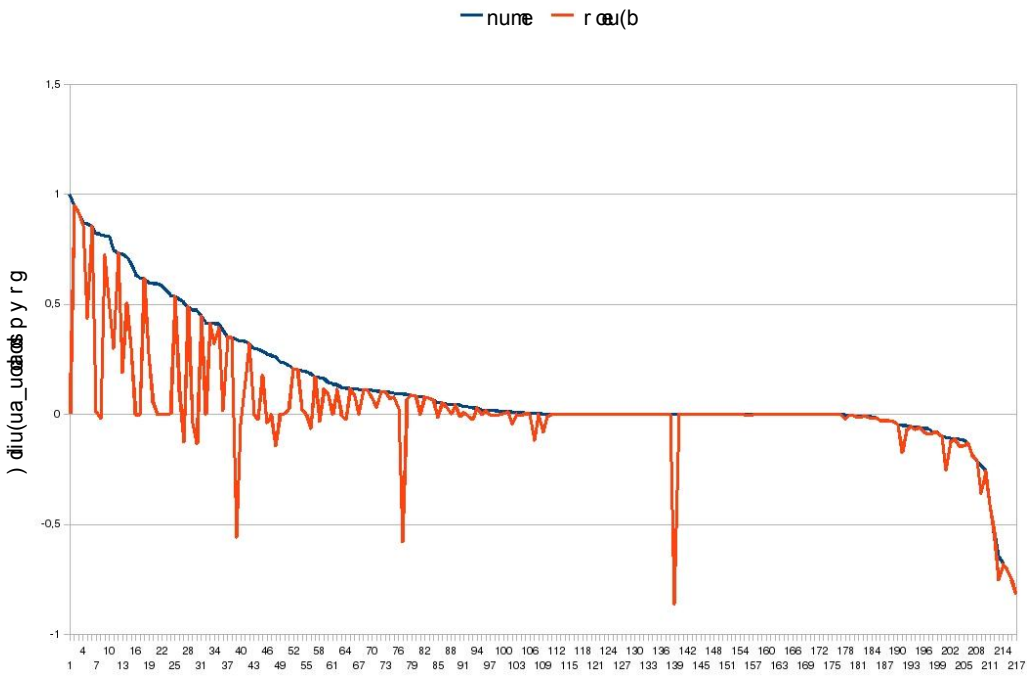


Figure 5.5: Differences in iP[0.01], respect to non-Relevance Feedback case, when we use the best number of terms and the best fixed number of terms (1), for the case $m = 5$.

number of terms increases. With respect to the part of the CAS query to be expanded, it is interesting to know which part of the query is more appropriate for expansion. The number of terms used for the expansion is ranged from 1 to 10 and the expanded parts are:

- *Full expansion*: All the subqueries of the structural query are expanded.
- *Context expansion*: Only the subqueries of the context part are expanded. In this case we shall considered both the soft and hard version of the non-relevant assumption.
- *Target expansion*: Only the subqueries of the objective part are expanded.

m	iP[0.01]	iP[0.05]	iP[0.10]	AiP
5	104	94	95	85
10	79	74	81	67
20	69	67	70	65

Table 5.10: Number of queries where RF using the best number of expanded terms obtains worse or equal results than without Relevance Feedback.

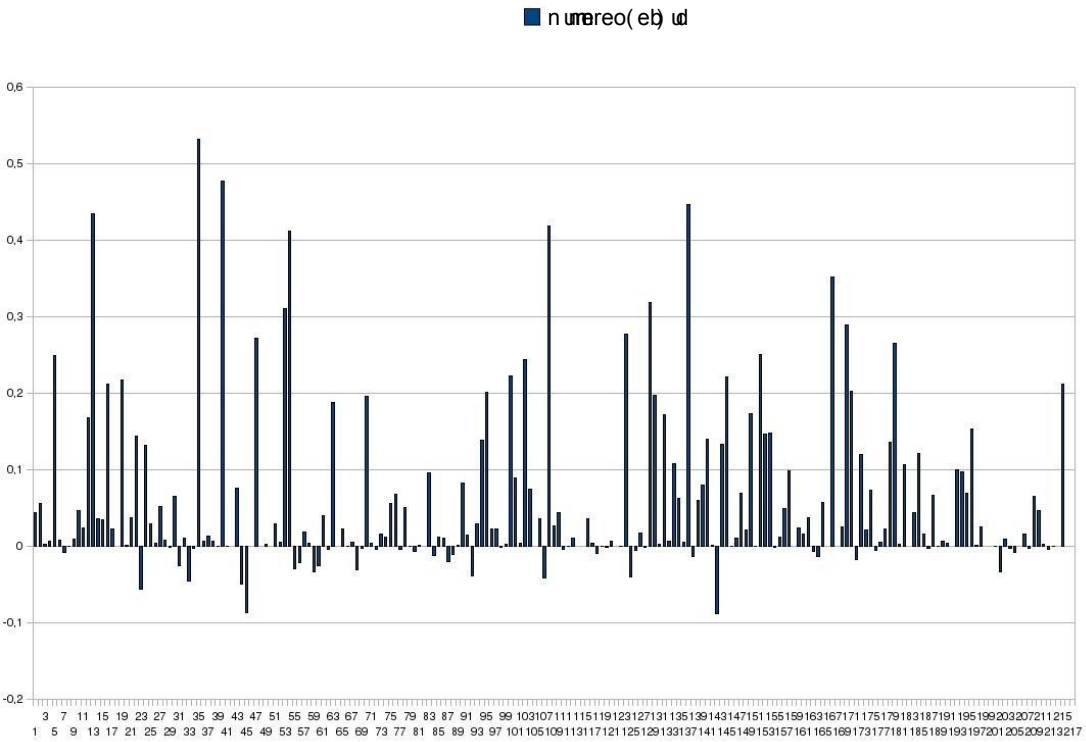


Figure 5.6: Differences between AiP values, respect to non-RF case, when we use the best number of terms, for all the queries and $m = 10$.

The results of these experiments, taking into account that the number of judged units is 10 ($m = 10$), are summarized in tables 5.11 and 5.12. They display the corresponding effectiveness measures for the baseline (*Original query*) and for each one of the experimental settings.

Focusing on the context subqueries (table 5.11) we use bold face fonts to indicate the best results for each measure. Also, in those columns associated with

	Context Exp (Hard).				Context Exp (Soft).			
k	iP[0.01]	iP[0.05]	iP[0.10]	AiP	iP[0.01]	iP[0.05]	iP[0.10]	AiP
without RF	0.3460	0.2589	0.2214	0.0831	0.3460	0.2589	0.2214	0.0831
1	**0.3539**	**0.2729**	**0.2427**	**0.0848**	**0,3323-**	0,2624-	0,2213-	0,0833-
2	0.3144	0.2652	0.2346	0.0822	0,3148+	0,2754+†	0,2326-	0,0842+
3	0.2998	0.2499	0.2292	0.0799	0,3044+	0,2709+	0,2376+	0,0848+
4	0.2990	0.2555	0.2319	0.0806	0,2907-	0,2667+	0,2373+	0,0838+
5	0.3131	0.2650	0.2382	0.0816	0,2939-	0,2717+	0,2444+†	0,0841+
6	0.3078	0.2639	0.2366	0.0802	0,3010-	0,2768+†	**0,2526**+†	**0,0849**+†
7	0.3147	0.2634	0.2338	0.0802	0,2956-	0,2719+	0,2484+†	0,0848+
8	0.3064	0.2642	0.2337	0.0794	0,3047-	**0,2793**+†	0,2369+	0,0826+
9	0.3143	0.2645	0.2315	0.0790	0,2920-	0,2666+	0,2326-	0,0832+
10	0.3167	0.2696	0.2287	0.0792	0,2948-	0,2617-	0,2267-	0,0825+

Table 5.11: Results of the experiments for the context expansion.

	Full Exp.				Target Exp.			
k	iP[0.01]	iP[0.05]	iP[0.10]	AiP	iP[0.01]	iP[0.05]	iP[0.10]	AiP
without RF	0.3460	0.2589	0.2214	0.0831	0.3460	0.2589	0.2214	0.0831
1	0.4239	0.3267	0.2848	0.0920	0.4080	0.3196	0.2716	0.0911
2	0.4361	0.3439	0.2923	0.0939	0.4107	0.3349	0.2744	0.0943
3	**0.4433**	0.3558	0.2984	0.0952	0.4177	0.3343	0.2730	0.0959
4	0.4356	**0.3648**	**0.3186**	**0.1028**	**0.4312**	0.3392	**0.2955**	**0.1013**
5	0.4407	0.3604	0.3116	0.0993	0.4294	**0.3437**	0.2895	0.0995
6	0.4137	0.3530	0.3098	0.0998	0.4063	0.3255	0.2847	0.0995
7	0.4245	0.3488	0.3111	0.0986	0.4147	0.3230	0.2784	0.0982
8	0.4158	0.3444	0.3106	0.0984	0.3907	0.3092	0.2790	0.0975
9	0.4188	0.3454	0.3059	0.0976	0.4008	0.3091	0.2733	0.0971
10	0.4283	0.3505	0.3078	0.1006	0.3938	0.3036	0.2707	0.0982

Table 5.12: Results of the experiments full expansion and target expansion.

the soft non-relevant assumption (right hand side of the table), we denote by the symbol + (or -) when under the same number of expansion terms the soft version is better (worse) than the hard ones and we use the symbol † to indicate that this option is better than the best results obtained using the context expansion for the hard version, i.e., using only one expansion term ($k = 1$).

From table 5.11 some conclusions might be obtained, first that by expanding only the context subqueries minor improvements can be obtained with respect to the baseline (even in some cases the expansion of only the context part is not a good solution). Thus, the best results obtained for each measure achieve the following improvements: 2.30% for iP[0.01], 7.88% for iP[0.05], 14.09% for iP[0.10] and 2.16% for AiP. Second, we might say that the performance of the soft version outperforms the results of the hard version when, under the same circumstances, we compare 1-to-1, but soft version barely outperforms the best

results of the hard version, obtained when adding only one expansion term. As general conclusion, it is better to consider the hard version for the non-relevant assumption. This is because soft version, whose best results seems to be obtained using 6 expansion terms, does not get, in mean, better results than the best hard version. Therefore, in the rest of the section we shall only consider the hard version.

Focusing on table 5.12, the results obtained by expanding only the target subqueries and also when expanding both subqueries, target and context (in its hard version), named full expansion, are presented. In the case of full expansion, the same number of expansion terms in all the subqueries has been used. In this table we highlight with bold face the best results obtained for each measure.

The first conclusion that can be obtained is that the results obtained in this table are much better than the ones obtained using context expansion. Comparing full and target expansions, the first one, in general, obtains better results in all the measures with significant differences although it is smaller in the AiP measure.

With respect to query expansion and weighting of the new RT terms, using from 3 to 5 terms for every subquery is the best solution because the improvements that can be achieved go from 19.57% to 23.57% for the AiP measure in the full expansion and from 19.77% to 22% in the target expansion, but these improvements are higher if we consider other measures like the iP[0.10] where the percentages are ranging from a minimum of 39.92% to a maximum of 43.90% in the full expansion and from a minimum of 28.57% to a maximum of 33.48% in the target expansion.

It is believed that this best performance for these numbers of terms is due to the short length of the subqueries (around two terms in average, with a maximum of four terms). So, if we consider few terms (1-2) we can get some improvements but they are not so significant and the use of more than 6 terms provokes the decreasing of performance. It is due to we could be introducing non-representative terms in the expanded subqueries, changing the focus of the original query and hence generating worse results.

Another important conclusion is that context and target expansion reinforce each other. In all the experiments we have that the measure's values of the full expansion are strictly greater than the ones obtained by expanding context or

target isolately. Moreover, the improvement's percentages obtained with a full expansion are greater than the sum of the improvements obtained by context and target expansion.

Once it is known that full expansion is the most appropriate approach to expand CAS queries, we could wonder how a different number of expansion terms in the context and target subqueries could affect the retrieval performance. In order to answer this question, the following experiment has been designed: Firstly, the number of expansion terms is fixed to 4 for the target subquery (we assume that a good performance is achieved with this configuration). Then, a new experiment has been run by varying the number of expansion terms in the context subqueries from 1 to 10. Table 5.13 shows the absolute results obtained (in boldface the best results for each measure).

k	iP[0.01]	iP[0.05]	iP[0.10]	AiP
without RF	0.3460	0.2589	0.2214	0.0831
1	0.4355	0.3591	0.3121	0.1035
2	**0.4427**	**0.3700**	0.3164	**0.1040**
3	0.4349	0.3643	0.3159	0.1019
4	0.4356	0.3648	0.3186	0.1028
5	0.4401	0.3681	**0.3202**	0.1015
6	0.4326	0.3633	0.3196	0.1004
7	0.4345	0.3626	0.3178	0.0995
8	0.4313	0.3568	0.3160	0.0994
9	0.4229	0.3461	0.3067	0.0972
10	0.4269	0.3510	0.3122	0.0979

Table 5.13: Results of the experiments varying the number of expansion terms (k) in context subqueries and fixed to 4 in target subqueries.

Observing this table, we could conclude that the consideration of a different number of terms used to expand in both parts of the CAS query is relevant in terms of retrieval performance. In general, the best overall results are obtained by using a different number of terms: four in the target (fixed) and two terms in the context, although it represents a minor improvement (around 1%) with respect to using four terms also in the context. These results agree with the ones obtained expanding only context in the original query, as a large number of expansion terms in the context seems to worse the performance. It seems that it is more interesting to leave almost unchanged the original context subqueries, being less important than the target subqueries for retrieval purposes.

5.6 Conclusions and future research

We present in this chapter two RF frameworks for Content-only (CO) and Content and Structure (CAS) queries, respectively. These approaches modify the original queries expanding the keyword queries in CO queries and keeping the same structural restrictions but expanding the keyword subqueries in CAS queries. In both cases, we select the best top weighted terms. These frameworks have been evaluated with INEX 2006 and INEX 2007 collections.

Studying the experiments of our Content-oriented framework, it is important to mention: (1) In general, RF improves the results for all the measures of retrieval effectiveness (using the residual collection method adapted for structured IR); (2) the results are better for measures focused on the recall; (3) they are better, as well, if we have more relevance information (there are many judged units); (4) With respect to the number of expanded terms, there exists a positive correlation with the number of judged units and negative with the wished recall level; (5) it is possible that the use of a variable number of expanded terms will improve the results; (6) there is an important percentage of queries where RF gets worse results independently of the number of terms.

Consequently, we are planning to study an automatic mechanism to determine in each query, depending on its features, if it is advisable to use RF, and in an affirmative case, how many expanded terms we should use. Also, it could be interesting to study how to apply negative RF, i.e. how to use the terms in the non-relevant units in order to best capture the real intent of the user query.

In our Content and Structured-oriented framework, our future work will include the objectives of the Content-oriented framework to expand the keyword subqueries. Another interesting issue could be to modify structural constraints (only in the context) in the new expanded query. The objective part should not be changed because this indicates the structural unit which the user is interested in. So, it must be the same in both original and expanded queries but the context restrictions, in some cases, could be interesting to change it.

Part IV

Applications to the Andalusian Parliament

Chapter 6

Description of the problem for the Andalusian Parliament

6.1 Introduction

With the development of computers and Internet, most organizations working with documents (in every sense of the word), in a physical format, i.e., paper, witnessed the first computer revolution whereby such documents were converted to an electronic format. This allowed advantage to be taken of new technologies for managing document collections and resulted in a faster and easier user access. Many organizations and institutions, therefore, started to produce digitally-formatted documents (for instance, in Portable Document Format or PDF) which were then made available to a wider public by means of the Web, and search engines with basic functionalities were incorporated to enable users to search and access these digital libraries.

A step forward could be taken, with the Internet still at the center but surrounded by a whole "gamut" of new technologies that work simultaneously with several types of media. Information access is now:

- Faster: Because of the improvement in the underlying communication technology.
- More accurate: As the user obtains information which is closer to his or her information requirements.

- Easier: Because less effort is required from the user.
- More diverse: The sources are not limited to text, a mixture of text, images, audio and video can now be incorporated daily to enrich an organization's digital library.

The management of digital libraries and the digital libraries themselves must therefore evolve towards a new "digital framework", and the institutions must take up the new technological challenge in order to remain up-to-date.

Such is the case of the Andalusian Parliament (AP, Parlamento de Andalucía), corresponding to the Southern Spanish autonomous region, which was established in 1982. One of the main objectives of democracy is that citizens are informed of any decisions made by their parliamentary representatives at any given time.

National and regional governments are consequently obliged to inform the public of any work undertaken by parliament, so that all the matters discussed are public knowledge. Initially, the AP published a record of parliamentary proceedings which consisted of documents printed on paper with exact transcriptions of all the speeches by Members of Parliament (MPs, Diputados) relating to every matter discussed. These bulletins were then sent to official organizations and public libraries in order to they were publicly available.

The first challenge for the AP in terms of technology was to create a digital library with all the records of parliamentary proceedings which users could easily access. Then, they changed their modus operandi and PDF documents were generated from the transcriptions. This file type is currently the most widely used in organizations for storing and spreading textual information as it enables electronic documents to be exchanged and viewed reliably and easily, regardless of the environment in which they were created.

Having prepared the e-documents, the organization included a search engine on its website[1], so that any Internet user (not just politicians or parliament employees) could use a form to submit a natural language or database query to obtain any relevant documents.

As an alternative to the manual transcriptions of the speeches, video recordings of the sessions were introduced to supplement the Parliament's available

[1] www.parlamentodeandalucia.es

information resources. Up until now, these two completely different media forms from the same source were dealt with separately: the user was either looking for text or a video but not both, intersynchronized with one query. This type of multimedia retrieval is one example of a challenge posed by the technological revolution.

A second interesting example relates to the format of the documents in the digital library and (as mentioned) PDF was the chosen format. When a user submits a query, the full document is returned, and the information required might be found in a certain paragraph or section of the document in a MP's speech. The user must therefore read the whole document to find the required information and it is a waste of time. This is the classic perspective on IR. However, if the official parliamentary document was extremely well organized with a rich internal structure, we could take advantage of this and, rather than returning the full document, the part that best matches the user's information requirements could be returned. The retrieval system will direct the user to specific parts of the document, which are relevant to their query and it supposes a save of time.

Since PDF is not the most appropriate format to store a document's structure, it is therefore advisable to select another document format and for these purposes XML is a very suitable option as it is able to capture the internal organization of the textual materials. It represents a move away from the classic field of IR to a new area of structured IR, which deals with explicitly organized documents, whereby the problem is not to retrieve a relevant document but rather relevant parts of it.

In this chapter, we present a case study for the AP and show the methodology that was designed and applied in the framework of a research project to improve the parliament's digital library in terms of its document collection and document access infrastructure (although there is no commonly-agreed definition, these are the basic components of a digital library according to [87]). This methodology was of course particularized to the specific problem in hand but the steps could easily be generalized and adopted by any similar organization.

Therefore, we present an integrated information system, called *Seda*, comprising various software modules which have been designed and implemented to

improve access to the documents and videos in the AP's repository by exploiting the documents' internal structure for retrieval purposes.

In the following section of this chapter, we provide some background information about the Andalusian Parliament and its publications. Section 6.3 analyzes the possible weaknesses detected with the digital library and possible solutions for strengthening it. Section 6.4 describes the methodology that we have designed for our case study and translation to a computer application (*Seda*). Moreover, it shows us the outline we have followed in the application part.

6.2 The case study: The Andalusian Parliament and its digital library

This section presents a brief introduction to the AP and its official publications in order to contextualize this chapter and introduce specific terminology.

As mentioned, the AP was established in 1982 and to date there have been eight legislatures (periods of political activity of up to four years). The Parliament edits two main official publications: *the record of parliamentary proceedings (diario de sesión)* and *the official bulletin (boletín oficial)*. The first publication contains full transcriptions of all the MPs' speeches in each parliamentary session in which laws are passed or in the informative sessions held with MPs. Additional information such as the official agenda (the matters to be discussed as agreed by all the political groups, the minutes of the meetings), results of possible votes, agreements, etc. is also included. In the second publication, the Parliament publishes any texts and documents to be made available to the public about laws passed or to be processed.

There are three main types of sessions:

1. *Plenary sessions (sesiones plenarias)*: these are attended by all MPs to debate an initiative.

2. *Committee sessions (comisiones)*: these are attended by MPs corresponding to different areas of interest (agriculture, economy, education, etc.) to discuss relevant initiatives.

3. *Permanent parliamentary sessions (sesiones de la diputación permanente)*: these are attended by various duty MPs when Parliament is not in session.

Parliament works around the concept of parliamentary initiative (iniciativa parlamentaria), whereby an action taken by an MP or political party is discussed in a plenary or specific area committee session. These initiatives are included in the plenary or committee sessions and identified by means of an initiative code (expediente). Before a plenary session is held, the political parties represented in the House decide on the agenda for the session and it consists of a sequence of initiatives. These are subsequently grouped according to type and are previously published in the official bulletin. Once the agenda has been agreed, the Speaker leads discussion of each point, allowing MPs to speak on and discuss the corresponding initiative.

All official documents are published electronically on the AP's website, and PDF is currently the format most widely used by organizations (including the AP) for storing and publishing textual information.

When the PDF documents have been published, the AP also provides a search engine in order to users can consult the legislative collection by means of a database-like query using a web form. As the sessions are also recorded, the Parliament's digital library with its records of parliamentary proceedings is supplemented with videos on the website which, while not able to be searched, can be browsed by date and viewed.

Since it was established 27 years ago, around 6000 documents have been published with the records of parliamentary proceedings and the official bulletins. While the size of this digital library is not excessive, it is a respectable figure and is constantly growing.

To help the reader understands the rest of the chapter, we shall now describe how the records of the parliamentary proceedings and the official bulletins are organized. The records of parliamentary proceedings comprise four distinct yet well-defined parts:

1. *General Information section*: it contains general information about the parliamentary session in question. For example, type of session, legislature, date and presidency.

2. *Agenda (orden del día)*: a list of the initiatives grouped according to type. It is decided by the political parties before the session and follows the format of initiative code, subject and proposer. For example, in the type "oral questions", all the questions are listed specifying for each one the question and the asker.

3. *Summary (sumario)*: a detailed description of the agenda which is created once the session has finished. Once again, all the initiatives are grouped according to type and in addition to a description of the initiative (code, matter and proposer) they also include a list of MPs participating in the debate and the result of any vote. New agenda items may be added or others removed.

4. *Development of the session (desarrollo)*: For each point included in the summary section (following the initiative information), transcriptions of all the speeches are included and set out like a script for a play or a film.

The official bulletin is organized in the following way:

1. *Summary*: In addition to identification and date of the issue, this includes a brief reference to each parliamentary initiative developed in the main body of the document and serves as a type of table of contents. There exists a well-defined and static hierarchical taxonomy of initiatives designed by the Parliament (for instance, law projects, oral and written questions, motions, etc.) so all the initiatives presented in each document are arranged in the corresponding place in the hierarchy.

2. *Body of the bulletin (cuerpo)*: Following the taxonomy presented in the table of contents, this develops the content of each initiative and includes information such as the initiative number, proposer, answerer (in the case of a question to Parliament, for example), date, and the text itself explaining the request or answer.

So that these documents can be accessed, the institution's web team maintain a search engine whereby users search one type of publication at a time, expressing a query in natural language as well as specifying the date of publication,

identification of the document and/or legislature (or ranges for this data) in a database-type query. The query returns a list of PDF files sorted according to publication date rather than relevance to the query. It is worth mentioning that these two types of documents present a very rich, well-defined internal structure, which has not been exploited for retrieval purposes until now.

6.3 Detected weaknesses and possible improvements

In technological terms, the Parliament's digital library (with its collection of documents and access methods) has certain drawbacks that need to be resolved in order to improve quality:

- As documents are searched in a database-type style, they are displayed in descending order of date and cannot be shown in terms of their relevance to the user's query (those most relevant, first).
- In order to find relevant information, users must select the type of document they want to search. If they want to search various types of documents, they must submit identical queries for each document type.
- Users introduce queries in which they can not indicate where to look for. So, if you introduce a query, the system looks for the terms of the query inside the whole documents. For this reason, this type of queries are not very specific.
- Documents are treated as a whole, i.e., as atomic units. Once a PDF file is retrieved, the user must search for the relevant information within the document, with the time-wasting that it involves for the user.
- There is no Relevance Feedback mechanism with the objective of improving the quality of the results of the original query.
- Videos are only searched by date; users must know the exact date of the session recording to locate the video they want to find.

- Users must watch the full video to find the relevant part with the information they require.
- There is no connection between the records of parliamentary proceedings and the videos. Although these two different types of media represent the same session, they are not interconnected. So, there is no direct access between a retrieved PDF file and its associated video.

Bearing these weaknesses in mind, and considering leading technologies, we believe that improvements in digital library management and use are possible, and therefore propose the following points:

- By applying IR methods and techniques, and given a query formulated in natural language, it would be possible for users to obtain a ranked list of all types of documents which are sorted according to relevance to the query (the higher the document in the ranking is, the more relevant). Therefore, it is only necessary for users to read the uppermost-ranked documents to find the information they require.
- Thanks to the document's internal organization, it is possible to apply structured IR techniques in order to the search engine could return a ranked list of document parts (rather than full documents) in response to a query. It is clearly an important advantage because the system directs the user to the exact text unit where the relevant information is found regardless of its size and it is not necessary for the user to waste time finding what they are looking for. The main requirement of this new approach is to represent the documents in a much more flexible format, where their structure is explicit and easily managed. XML is the most suitable option for such a task and this meta-language has been used extensively in digital libraries (see [58], [106], [14] for examples). Additionally, the search engine should be endowed with specifically designed retrieval models for managing structured documents and retrieving parts of them.
- A powerful query mechanism that would enable the user to formulate more accurate queries to help them find the information they require. In addition to natural language queries (with possible restrictions on the type

of document, legislature, date, etc.), the user could specify what to look for (content), where to search, and the required output. In the context of structured IR, it is called a *Content and Structure query* (CAS queries) and offers the possibility of specifying which parts of the documents to search and which to return.

- Improving the quality of the results given by the search engine using *Relevance Feedback* (RF) techniques. In this sense, the user could decide which results of the search engine are interesting for the original query, and then the system would generate a new query adapted to the user's information need using the original query and the judged results. Finally, the search engine would retrieve results closer to the user's information needs.

- A link between the records of parliamentary proceedings and their associated videos, so that users could read the most relevant parts of the documents and watch the section of the video corresponding to the relevant part of text. Not only does the user obtain various supplementary multimedia sources in response to a single query, but they are also directed to the relevant part of the video with the purpose of they do not waste time trying to find it themselves in the full video. In order to establish such an association, additional tools must be developed and these include an application to segment videos and an annotation tool to synchronize text and video. We should highlight that with this connection, videos are retrieved and attached to a portion of text as a result of a query, unlike other specific and more complex video retrieval techniques (see [75]).

By developing an information system for the Parliament's digital library, we intend to solve some of the problems detected, and facilitate internal processes and access to the official collections, making these more efficient and effective.

6.4 Methodology and system architecture

In this section, we shall describe the methodology that we have designed and followed to technologically update the AP's digital library. Although this methodology has been particularized to the specific problem in question, it could easily

be generalized and exported, so that it may be applied and adapted to any other institution or organization with similar requirements. Our final aim is to develop an information system that covers the whole process of edition, publication and access of the elements comprising the digital library.

In addition to explaining the methodology, and as a consequence of its application, we shall also show the software infrastructure of the general information system (*Seda*) in terms of the required modules and relationships between them.

The following steps have been taken in order to satisfy the general requirements presented in the previous section:

1. *Design of a Document Type Definition (DTD)* to represent the internal structure of the records of parliamentary proceedings and official bulletins. The two types of official documents must be analyzed to extract the exact internal structure governing them. This is a manual process which results in a DTD file which contains an explicit description of the document's structure.

2. *Document conversion from PDF to XML.* As the original documents are in PDF format which does not usually store information about a document's logical organization and since the IR system will work explicitly with the structure and content by means of XML documents, it is necessary to convert the formats automatically.

3. *Video segmentation and synchronization with the text.* In order to link a part of the record of parliamentary proceedings with its corresponding part of the video (for example, the text of an MP's speech and the speech itself in the video), it is necessary to split the video into segments. After this semi-automatic segmentation process, the parts of the text and the segments for the same speech must be associated (synchronized). This is a mainly manual but computer-assisted process.

4. *Design of a query mechanism.* The way users express their information requirement is very important. The different alternatives (natural language, CAS, restrictions on legislature, dates, type of document, etc.) call for the design of a complete, intuitive mechanism to facilitate this task.

5. *Design of a retrieval model for structured documents and development of the corresponding IR system.*

6. *Design of a Relevance Feedback mechanism.* The results of the original query are evaluated by the users and the relevance assessments as well as the assessed portions of documents are used to generate a new query more adapted to the users' need.

7. *Preparation of the document collection (indexing).* In order to access the document collection by means of queries, the IR system must adapt the XML documents to its internal data structures, so that they may be effectively and efficiently accessed.

8. *Design and development of a graphic user interface and integration of the search engine.*

The application of these steps results in the software modules shown in fig. 6.1, which shows the different application elements and the relationships between them in terms of inputs and outputs.

Beginning with the PDF collection, one of the first modules to apply is the PDF-to-XML converter. It will transform all the PDF documents to XML format by extracting the text and placing it in the correct position in the structure of the record of parliamentary proceedings and the official bulletin.

In parallel, each video of the parliamentary proceedings will be automatically segmented by the video segmentation tool to obtain a series of segments marked by a start and end time. These video portions could be manually edited to adjust the original output given by the software.

Once both processes have been completed, the records of parliamentary proceedings and their recordings must be synchronized. In this case, given an XML record of parliamentary proceedings, time attributes pointing to the video segments are included in the XML tags associated with the speeches. A time propagation is then performed towards the upper elements of the document structure.

At this moment, and once the search engine has indexed the XML collection so that it may be managed efficiently, the user is able to formulate a query on the web-based user interface. This query is passed to the search engine which

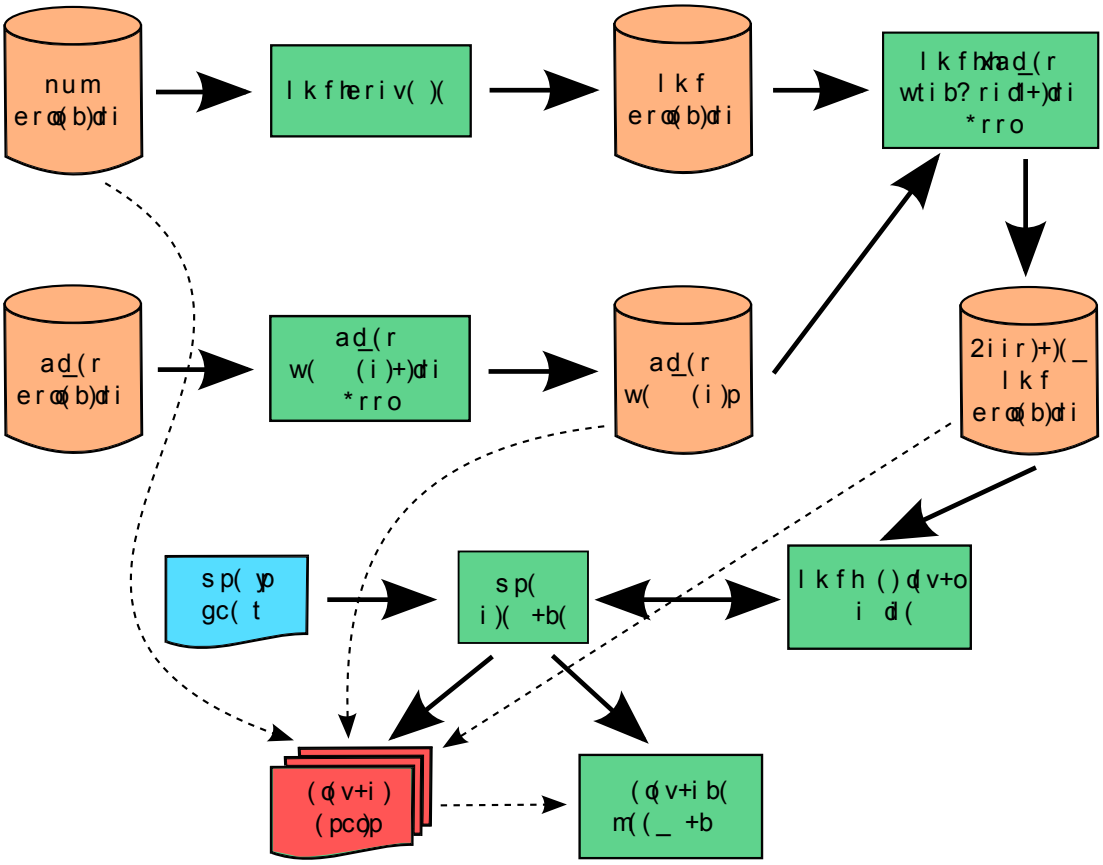


Figure 6.1: Architecture of *Seda*.

then computes the most relevant XML elements using a retrieval model based on Bayesian networks and influence diagrams and implemented in the Garnata IR system. These elements are ranked in decreasing order according to relevance and shown to the user, and can be arranged in various different ways (for instance, all the results, grouped according to document, (*relevant in context*) or the best entry point for each relevant document (*best in context*)). Then, users can view the XMLs, PDFs and videos to find the information they are looking for.

In the following chapters, we shall describe the specific modules developed in this project for this case study, so in the chapter 7, we shall see the creation of the XML collections of the AP. Chapter 8 will show the way used to segment and synchronize the videos. Lastly, in chapter 9 we shall describe all the web

interfaces developed for *Seda* to introduce queries, show relevant results, etc.

Chapter 7

XML collections of the Andalusian Parliament

7.1 Introduction

At present, the Portable Document Format (PDF) is used in most of the organizations, included the AP, to store and spread textual information, since it enables to exchange and view electronic documents in a reliable and easy way, independently on the different environments in which they were created and viewed.

Once the PDF documents are published, the AP provides a search engine by which users can consult the legislative collection by means of a query in natural language (as we could do with any search engine as Google, for example). The result of this process, where techniques provided by the IR discipline [8] are applied, is a set of PDF documents, sorted according to publication date, which try to fulfill the user's information need. The user then clicks in each one of them in turn and tries to find where her/his information need is satisfied inside the document.

If we take into account that all the official documents of the Parliament (records of parliamentary proceedings and official bulletins) present a very well defined structure, as well as the fact that each record of parliamentary proceedings contains an exact replica of its corresponding session, the content of each PDF is organized according to a strict and rich structure that may be useful in terms of retrieval.

In this chapter, we present the XML collections of the AP for both types of documents because XML format permits us to represent content and structure as we commented in section 1.3.1 of chapter 1. In concrete, the conversion from PDF to XML of the records of parliamentary proceedings has been one of the objectives of this thesis, however the creation of the collection of official bulletins was developed in another project, so we focused on the development of a software converter specifically designed for the specific nature of the records of parliamentary proceedings of the AP, but designed under a robust methodology which may be exported to any other type of document that contains an intrinsic structure.

The aim of the converter is to try to find the elements of the structure in the PDF document, translate them into XML tags, and attach the corresponding text, giving as a result an XML file which is an accurate representation of the content and structure of the PDF document. This process is not usually direct due to a wide variety of peculiarities of documents belonging to different legislatures. Also, there are lots of exceptions that must be taken into account to produce a correct conversion.

The approach given by most of the existing general converters is rather different from ours, as they offer as a result a structure extracted from the document but related to the visual layout of the pages, i.e., they usually organize the output XML file around pages, paragraph, fonts, size of fonts, and modifiers of the fonts (bold, italics, etc.). But this information is not what we are looking for from our retrieval purposes. This is the reason why we did not use any of them and thought about developing a new one totally adapted to the features of our problem.

Our converter is composed of different components that cooperate in a sequential pipe (the output of one module is the input of the following). These are the following: a text extraction module to generate text files from PDF documents, a lexical analyzer, a syntax analyzer and an XML generator using DOM methodology.

The chapter is organized in the following way: In section 7.2, we show a general description of the problem to know the purpose of our work. Section 7.3 explains the structure of the records of parliamentary proceedings and the DTD file. Section 7.4 describes all the conversion process from PDF documents to

XML files. Section 7.5 does a comparison between other works similar to this one and our work. Section 7.6 presents some information about the XML collections like the DTD file of the official bulletins and section 7.7 includes the conclusions and some remarks about further research.

7.2 Description of the problem

The AP has generated a large set of PDF documents since its origin, and although these PDF documents have a well defined structure in their contents because they follow a clear order (general information about the session, agenda, summary and session development), this content-internal structure is not enough to use it in a structured IR system. The reason is that we are only able to extract plain text without any additional knowledge that tells us if a sentence extracted from the text is a common paragraph or the title of a section, for example. It is the knowledge that we want to get. Of course we can get also typographical information, as type of fonts, sizes, and so on, but this is not interesting for our purposes.

When the corresponding administrative department of the Parliament decided to publish the records of parliamentary proceedings in the Web, they used the most common format to do it, the Portable Document Format, but at that moment, they did not know that they could take the most of additional information (the structure) from a perspective of a search engine. It could have been very useful in order to help the users to get relevant material. So, the problem is to generate an alternative representation of the records of parliamentary proceedings that allow feeding the search engine for these purposes. This is the reason by which we need software that is able to extract the text contained in all the PDF documents composing the legislative collection and organize it according to its logical structure.

The transcriptions of the parliamentary sessions produce PDF documents called records of parliamentary proceedings containing the transcription of the sessions plus some additional information. In order to extract the text contained in the files, we have two options: To use an intermediate tool that gets a text file, which will be the input of the XML converter, or to extract the text by means of

a toolkit that allows accessing the PDF but totally integrated in the developed software. We have opted by the first choice: an external PDF to text converter is used, specifically, *pdftotext*, whose output will be the input of our XML converter. Afterwards, our application generates the final XML file starting from the text files with the textual content of the PDF. The software has been developed in Java, and performs the following steps:

1. By means of a lexical analyzer, it processes the text of these files and produces, as output, a sequence of symbols called tokens. In order to do that, the system introduces in the text a group of structural components or labels which correspond with the tokens, indicating the bounds of the different sections found in the document, as the types of matters discussed, the initiative code, the participations of the members of Parliament, etc. Another task performed by the lexical analyzer is the noise elimination, i.e., the deletion of those parts of the document which are not important from a semantic point of view. These parts are the information about other publications that appears at the end of the documents or the headers of the pages. Finally, the lexical analyzer places in the right position the few parts of the documents that the *pdftotext* converter disorganizes (as the plain text does not reflect the layout of the document, headers and footers are placed directly inside the text); therefore, these parts are always the same so it is a very easy problem to solve.

2. When the lexical analyzer has finished, the converter runs a syntax analyzer developed in *javacc*, in charge of generating a grammar to detect the tokens commented in the previous step. There are two main reasons of using *javacc*: it is free software and the most popular parser generator for using with Java applications.

3. For the creation of XML files, we use the DOM methodology based on the building of a tree in memory whose nodes are XML tags. Therefore, when a token is detected a new node or a group of nodes are created in the DOM tree, generating all the hierarchical structure of the XML format. The DOM tree is completed when the syntax analyzer has finished. Then, this tree

is converted into the corresponding XML file. For each document a new DOM tree is generated.

4. At the end, when we have all the individual XML files of the records of parliamentary proceedings, another Java application is executed to generate a general and unique XML file which contains the references to all the XML files of these records organized in the different legislatures.

5. Finally, to check if the general XML file is valid and well-formed, a validation process is launched and a message is shown indicating if it is correct. In case of the validation is wrong, the problem is indicated too. An example of a record of parliamentary proceedings in PDF format converted into an XML file is shown in fig. 7.1 and fig. 7.2. They show the conversion of the general information section. In fig. 7.1, we can distinguish the different components of the structure of this section in red boxes, and in a red font, the tag names. Fig. 7.2 shows the corresponding portion of the XML file with these tags and the information stored in them.

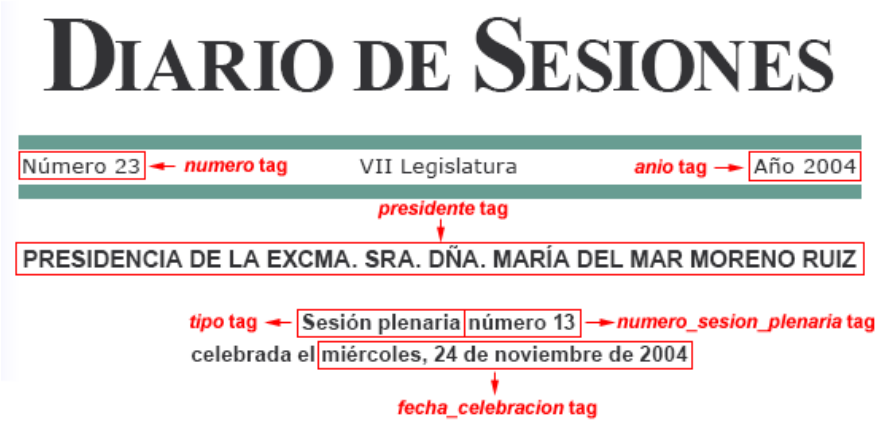


Figure 7.1: General information section.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <diario_sesion_pa tipo="plenaria">
    <anio>2004</anio>
    <numero>23</numero>
  - <pleno>
      <presidente>PRESIDENCIA DE LA EXCMA. SRA. DÑA. MARÍA
                  DEL MAR MORENO RUIZ</presidente>
      <numero_sesion_plenaria>13</numero_sesion_plenaria>
    - <fecha_celebracion>
        <dia_semana>miércoles</dia_semana>
        <fecha_dia>24</fecha_dia>
        <fecha_mes>noviembre</fecha_mes>
        <fecha_anio>2004</fecha_anio>
      </fecha_celebracion>
```

Figure 7.2: General information section of a record of parliamentary proceedings in XML format.

This is a general overview of the methodology used in our work to solve the problem. But we can actually find a lot of situations in which companies or government services have a great amount of PDF documents which are not structured themselves, but putting into practice a similar methodology, adapted specifically to the exact situation or context, a structured set of documents could be obtained, facilitating then the possibility of applying a structured IR system.

7.3 Structure of the XML records of parliamentary proceedings files. Designing the DTD

In order to understand the rest of the chapter, the description of the internal organization of the records of parliamentary proceedings is going to be explained. This analysis, much deeper than the presented in section 6.2, as we have to consider any single case or exception found in each document, will lead to the design of a DTD file containing an accurate representation of the structure that all the XML files must follow. In this type of files the hierarchical structure to be used is defined using a specific syntax, declaring the name of the tags that will appear in the XML files. The DTD file has the following structure:

1. The *diario_sesion_pa tag* contains the record of parliamentary proceedings information like the legislature (*tag legislatura*), the year (*tag anio*), the number of the record (*tag numero*), the serie if it is a comission (*tag serie*), and several tags which are stored in the *tag organo*: type of session (*tag tipo*), president of the session (*tag presidente*), number of plenary session (*tag numero_sesion_plenaria*) and the date of the session (*tag fecha_celebracion*). The fig. 7.1 shows all these tags and we can see the portion of DTD corresponding to this section in the following paragraph.[1]

```
<!ELEMENT diario_sesion_pa (legislatura?, anio, numero, serie?, organo)>
<!ELEMENT organo (tipo, presidente, numero_sesion_plenaria?,
fecha_celebracion, orden_del_dia?, sumario, desarrollo)>
```

2. Agenda: The agenda, i.e., the list of those initiatives to be discussed a priori (*orden_del_dia tag*), is also stored in the *organo tag*. It is formed by several *tipo_iniciativa* tags, which are the types of initiatives, or the initiatives without indicating the type (*iniciativas tag*) scheduled to be treated in the session.

   For each type of initiative, we store the following information: the title (*punto tag*), and a repetition of one of two types of tags or a combination of both of them: if it is a grouped debate, i.e., several motions discussed under the same point of the agenda (*debate_agrupado tag*), it contains its description (*descripcion_debate_agrupado tag*) and the initiatives of this grouped debate (*iniciativas tag*). If it is an individual initiative (*iniciativas tag*), its title (*tipo_expediente tag*), file number (*numero_expediente tag*), summary of the matter (*extracto tag*) and the members of parliament who present this topic (*proponente tag*). Fig. 7.3 shows this structure.

```
<!ELEMENT orden_del_dia ((tipo_iniciativa | iniciativas)+)>
<!ELEMENT tipo_iniciativa (punto, (debate_agrupado | iniciativas)+)>
<!ELEMENT debate_agrupado (descripcion_debate_agrupado, (iniciativas
| debate_agrupado)+)>
<!ELEMENT iniciativas (tipo_expediente?, numero_expediente?, extracto?,
proponente? )>
```

[1] All the tags which appear in the portions of DTD and are not defined in this chapter are #PCDATA.

Figure 7.3: Agenda of a record of parliamentary proceedings.

3. Summary: The summary (*sumario tag*), placed into the *organo tag*, contains the date and the time when the session starts (*hora_inicio and fecha_inicio tags*) and finishes (*hora_fin and fecha_fin tags*), and the content of the summary (*contenido tag*). If the agenda is a schedule a priori of the session, the summary is the actual agenda which was dealt with.

   Below this tag we can find the list of the topics of the agenda discussed in the session (*titulo tag*). Inside this tag the description of the corresponding item (*descripcion_titulo tag*) is included. All the matters (*iniciativa_sumario tag*) discussed by the members of parliament are placed into *titulo tags*, although there could be some *iniciativa_sumario tags* which do not belong to any *titulo tag* and are stored inside *contenido tag*.

   Iniciativa_sumario tags store all the information about the summary of the initiative (*extracto_sumario tag*), the members of parliament who discussed about these initiatives (*intervienen tag*), the results of the vote (*votacion tag*) and if the initiative is discarded, postponed or is only a proposal (*incidencia tag*). Another tag is error rectification (*rectificacion_errores tag*)

which could appear inside the summary at the end, when a mistake of any type is detected and is corrected. We can see this part of DTD in fig. 7.4.

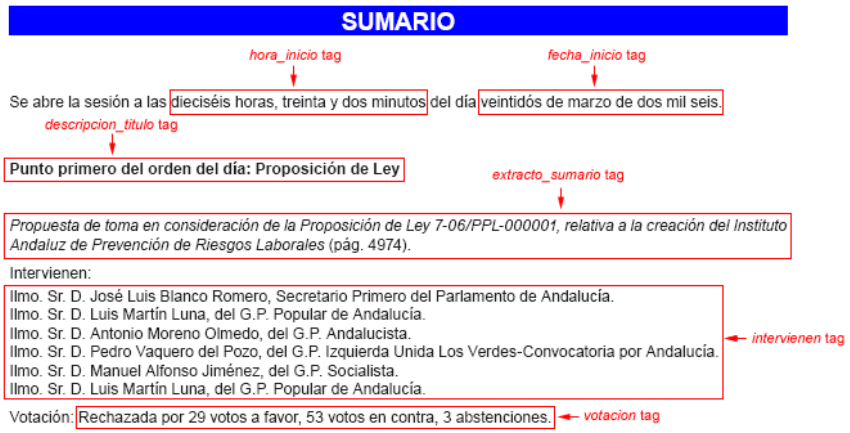


Figure 7.4: Summary of a record of parliamentary proceedings.

```
<!ELEMENT sumario (hora_inicio, fecha_inicio, contenido, hora_fin?,
fecha_fin?, rectificacion_errores?)>
<!ELEMENT contenido (titulo | iniciativa_sumario)+>
<!ELEMENT titulo (descripcion_titulo, iniciativa_sumario+)>
<!ELEMENT iniciativa_sumario (extracto_sumario, intervienen?, votacion*,
incidencia?)>
```

4. Development: The last part of the DTD is the *desarrollo tag*, stored in the *organo tag*, which contains the transcriptions of the discussions organized around the initiatives (*iniciativa_desarrollo tag*), its content consists of: the topic in which the initiative is classified (*materia tag*), the title of the initiative (*extracto_desarrollo tag*) and the *participacion_desarrollo tag* which contains the different speeches about the initiatives. So, it is really similar to the script of a play or a film, who talked, and what she/he said, i.e., (*intervienen_desarrollo and intervencion_desarrollo tag*); the transcription

of the speech is organized in different paragraphs (*parrafo_desarrollo tags*). In fig. 7.5 we can see this fragment of DTD:

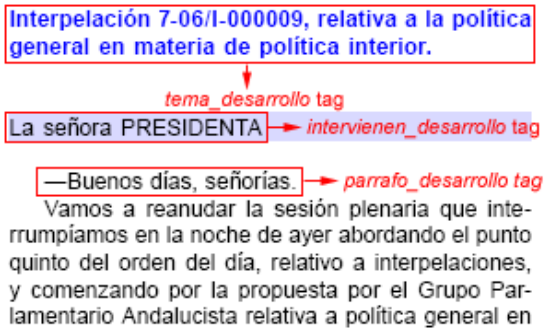


Figure 7.5: Development of a record of parliamentary proceedings.

```
<!ELEMENT desarrollo (iniciativa_desarrollo+)>
<!ELEMENT iniciativa_desarrollo ((materia?, extracto_desarrollo,
participacion_desarrollo+)|((materia?, participacion_desarrollo)+))>
<!ELEMENT participacion_desarrollo (intervienen_desarrollo,
intervencion_desarrollo)>
<!ELEMENT intervencion_desarrollo (parrafo_desarrollo+)>
```

The hierarchical tree-like structure, our general XML file must follow, is described in the DTD file, which consists of a set of rules or declarations that specify which tags can be used in a document, and what they contain. It is shown in fig. 7.6.

7.4 The conversion process

7.4.1 General view of the process

The conversion process is composed, in general, of the following steps: Firstly, we want to obtain the text files for all the documents of the collection, but here we must distinguish between the documents from the VI and VII legislatures, which are in PDF format and they are transformed using a converter called *pdftotext*,

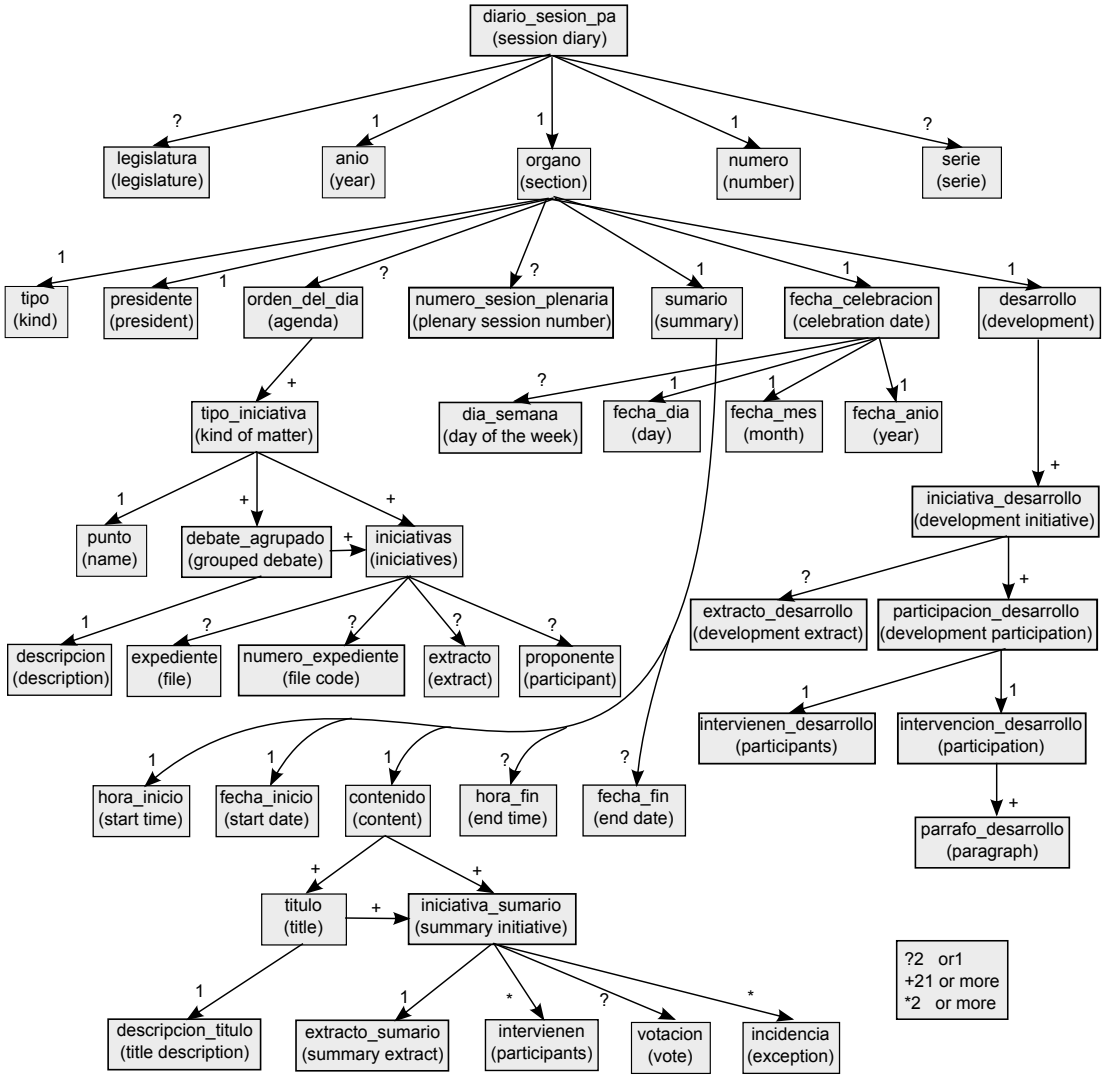


Figure 7.6: Tags structure of a record of parliamentary proceedings.

and the documents from the VIII legislature which are stored in both PDF and text formats, so it is not necessary to convert them.

Afterwards, the system pre-processes the text of these files with the aim of preparing it to identify the different parts (structural elements) of the document, deleting some parts of them which are unnecessary because they are not important from a semantic point of view, and adding some labels to indicate the content

bounds of the future XML tags. This task is performed by a lexical analyzer. Later, a syntax analyzer, developed in *javacc*[1], generates a DOM tree following the DTD, finally creating the XML document. To check if the XML file has been well-formed and its structure is correct, it is validated by the DTD file.

7.4.2 Text extraction

The text extraction task, developed for the documents from the VI and VII legislature, is performed by an intermediate, free converter called *pdftotext*[2] which converts the PDF documents into text files. It belongs to the *Xpdf* library which also contains other converters from PDF to other formats. One of the advantages is that it is multiplatform. *Pdftotext* extracts text objects from the PDF documents. Word and line segmentation is produced based on heuristics using the distance between characters and their geometrical positions.

This converter is easy to use and generates a clear format so we made the decision of using it, although we studied other converters like *Able2Extract* or *ConvertDoc* which have some disadvantages from our point of view, as they are commercial (not free), their text formats are less flexible, or they have more complex interaction. *PdfBox*[3] is a Java library to convert PDF documents into text too, it works properly but we discovered it could have some problems as we shall comment in the Related Work section.

The *pdftotext* converter generates the text files using the following command: "*pdftotext -raw -nopgbrk name_of_pdf_file name_of_text_file*". The parameter *-raw* keeps strings in content stream order and *-nopgbrk* does not insert page breaks between pages. After that, we obtain the plain text files with txt extension. Taking into account that the records of parliamentary proceedings in PDF are formatted in two columns, an interesting feature for us is the possibility of obtaining the text without considering this format, i.e., not observing the original layout of two columns as output. Therefore, the converter extracts firstly the whole text contained in the left hand side column writing it in the file, and secondly, it repeats the process for the second column (the right hand side). Some of the considered

[1] https://javacc.dev.java.net/
[2] http://www.foolabs.com/xpdf/
[3] http://www.pdfbox.org

converters introduce a line of the first column and the corresponding line in the second column in the same line of the output text file. The processing of this file in such a format is rather difficult.

7.4.3 Lexical analyzer

The lexical analyzer is the most important part in our system because it performs all the necessary changes in the text with the aim of facilitating the work of the syntax analyzer, which will be in charge of creating the XML file by detecting the corresponding labels. Basically the lexical analyzer searches the text using clues that give information about the structure of the text and include labels in the text that will help the syntax analyzer works later.

Once the text files of the collections have been obtained, the text is pre-processed by a lexical analyzer developed in Java. In general, the objective of this analyzer is to prepare the text to detect the different parts of the document easily. To perform this task, the lexical analyzer deletes all the parts of the document which are not important from a semantic point of view (noise elimination). These are the following parts:

- Several documents present advertisements about other Parliament official publications and subscription sheets, which are usually located at the end of the document. This information is not necessary for us, so the analyzer deletes it. It looks for a group of patterns in the development section which indicate the end of this part. When the system finds the furthest one, it eliminates all the information located after this pattern.

- Unnecessary pieces of information are the headers and footers, although the analyzer keeps the former delimiting them with special marks because they are useful in the further steps. To detect the headers, we have observed a lot of documents, finding a few types of headers. So, this task is very simple because the system only has to look for the patterns of all these types of headers in the text and deletes them. With the footers, the process is a bit different because the analyzer searches patterns of the type "\nNumber\n" (the number is the page number). Once they are found, it adds a label to

mark them. This mark is necessary to be detected by the syntax analyzer, as will be explained later.

- We noticed that the *pdftotext* converter disorganizes some parts of several documents when we get the text files (it misplaces some lines in wrong positions of the plain text) but it always happens in the same parts of the document. So, the system only has to search these parts in the documents and if they are not in the correct position the analyzer re-organizes the text putting them in the right place.

- There are some exceptions that are typical of few documents because they are mistakes of the PDF documents and are very difficult to detect. They are considered as special cases.

Once we have the text in a correct way for our purposes, the analyzer begins to add labels to indicate the content bounds of the XML tags, i.e. detecting the structural components that the text contains. In the following paragraphs, the labels which are introduced in the text are explained.

Labels added in the Agenda section:

- Firstly, the analyzer marks the types of initiatives that appear in the agenda. In order to do that, it needs the help of an auxiliary text file from the same PDF document but in a slightly different format generated also by the *pdftotext* converter using the parameter *-nopgbrk*. The objective is that the analyzer could find these types of initiatives easily since it is very difficult to detect them in the first text. In fact, our original text file does not keep several consecutive carriage returns (deleted with the *-raw* parameter) which usually appear before the type of initiative, being clues to detect them, although the auxiliary file keeps the carriage returns solving the problem. When the analyzer has found all the types of initiative in the auxiliary text file, it searches them in the original text and marks them.

- Another task is to mark the end of the initiatives in the agenda because it is an important task to delimit clearly all of them. To do that, the analyzer looks for paragraphs ending with ".\n" (full stop followed by a carriage

return), taking into account some special cases which are not the end of a paragraph. For example, "Sr.\n" (Mr) indicates the title of a person whose surname appears after. If the analyzer finds an initiative code in the following paragraph of this pattern or some specific type of initiative which do not contain an initiative code. Then, this pattern indicates it is the end of the previous initiative, so it places the corresponding mark.

- The people who are going to talk about a matter in the agenda is marked searching patterns like "formulada por" (formulated by) in every initiative. Therefore, we know that the text from these patterns until the label of end of initiative contains the different names of the people who took part in the debate.

- The initiative codes have a specific structure. For example, 7-06/PL-000086 is the identifier of an initiative related to a Law Project, with date July of 2006, and number of order 86. To detect them the analyzer uses a regular expression based on this structure. In spite of this, we found some initiative codes that are incorrect in the documents due to the special characters like "-" or "/" above all, which are replaced by other characters in the original files. So, the system fixes these problems searching strings with a pattern very similar to the initiative codes. Therefore, this pattern replaces "-" and "/" characters with any special character (no alphanumeric character), and in some cases, the incorrect characters with those previously mentioned.

- There are some initiatives which do not have a code to identify them. So, we have to deal with them in a different way in the syntax analyzer. As a consequence, the analyzer is not able to find an initiative code in these initiatives, adding a label at the beginning of these matters to distinguish them.

Labels added in the summary section:

- The system adds some labels to mark the time and the date when the session starts and finishes. This task is performed by looking for some common sentences that indicate the beginning or conclusion of the session. Therefore, we know that the date and the time go after these strings.

- Detecting in the summary section the people who are going to talk about a matter and its corresponding vote are the following tasks. In both of them the analyzer only has to look for very specific patterns and when it finds them, it adds the corresponding labels. Another task performed in the same way is to detect if a matter has been discarded, postponed or is only a proposal.

- After these last tasks, we want to mark the bounds of the initiatives and in order to do that the system looks for them in the paragraphs between every pair of labels added in the last steps previously explained. We have to take care in this task since the page number, the names of the people or the results of a vote appear in the same paragraphs. So, the system has to control them using some patterns to detect the initiatives correctly.

- The detection of error rectification is a very easy task because the system only has to search the pattern "\nRECTIFICACIÓN DE ERRORES\n" and mark it.

Labels added in the development section:

- We want to mark the people who talk in the debate and their participations. To develop this task, the analyzer looks for the patterns "El señor" (Mr.) and "La señora" (Mrs.) in the development section and marks them indicating the person who is going to talk. After its name, which is in upper-case (this is the clue to know when the name finishes), its participation (transcription of the speech) appears, which is marked too, following the organization of its paragraphs.

- The last task is the detection of the initiatives in the development section. This task requires the initiatives of the summary, previously saved in a vector, to look for them in the development section. The process should be easy: to look for each initiative, found in the summary, in the development section, but it is possible that they are not the same text because of some mistakes coming from the PDF to text conversion. So, we use the distance

of Levenshtein[1] to compare them and if the distance is less than a threshold we mark the text as an initiative in the development part.

After all these tasks, we have the text prepared to be analyzed by the syntax analyzer and detect the corresponding tokens.

7.4.4 Syntax analyzer and XML files creation

The syntax analyzer is developed in *javacc*. This parser generator reads a grammar specification and converts it into a Java program that can recognize matches to the grammar. This grammar is specified by studying the structure of the text files and the labels added in the previous step.

The analyzer works in the following way: It looks for the tokens, which are the labels we have used in the lexical analyzer to mark the text, following the order of the PDF document from the general information section to the development section. When it finds a token, the method related to that token is run, generating all the necessary nodes for the DOM tree, corresponding to the tags of the XML file. The text which appears after the label is stored in the corresponding tag. Therefore, the names of the tags and the parent tags are necessary to create the nodes in a correct way and place them in the right position.

Before the syntax analyzer begins to work, the constructor of the DOM tree is called, and a DOM tree is generated for every PDF document. When the syntax analyzer finishes, the DOM tree is converted into an XML file by the parser.

In fig. 7.7, we can see an example of this grammar for the agenda section. The token "<ORDENDIA>" appears firstly and contains the string "ORDEN DEL DÍA" (Agenda). In case of finding this token, a node for the DOM tree would be created with the name "orden_del_dia", whose parent node has the name "organo".

Afterwards, we could see a group of types of initiatives which are marked in the following way: The token "$I" indicates the beginning of the type of initiative and the variable "titulo" stores its name. When the analyzer has stored this name, the system creates the following nodes: Firstly, one node with the name "tipo_iniciativa", whose parent node has the name "orden_del_dia", another node

[1] http://www.merriampark.com/ld.htm

```
void OrdenDia():
{
    Token T;
    String titulo;
    String tituloCabecera;
    String expediente;
    String contenido;
    String proponente;
    String orden;
}
{
    <ORDENDIA>
    {
        crearNodoElement("orden_del_dia", null, null, "organo");
        System.out.println("Elemento orden del dia creado...");
    }
    (LOOKAHEAD(3)
    ("$I"
    (titulo=TituloCabeceraProponente()) {  crearNodoElement("tipo_iniciativa", null, null, "orden_del_dia");
                                           System.out.println("Elemento tipo iniciativa creado...");
                                           crearNodoElement("punto", null, null, "tipo_iniciativa");
                                           System.out.println("Elemento punto creado...");
                                           crearNodoText(titulo, "punto");
                                           System.out.println("El titulo es: " + titulo);
                                        }
    "$F")
    (LOOKAHEAD(4)
     ("<PAG:" <NUMERO> ">")
     | "*"
     | Debate_Agrupado()
     | Extracto_Con_Expediente()
     | Extracto()
    )+
```

Figure 7.7: Grammar corresponding to the agenda section.

with the name “punto”, whose parent node has the name “tipo_iniciativa” and finally, a text node which contains the name of the type of matter stored in the variable “titulo”, whose parent node has the name “punto”. To indicate the end of the type of matter, we find the token “$F”.

After each type of matter, there could be a group of matters of the following types: Debate agrupado (Grouped Debate), Extracto con expediente (Initiative with initiative code) and Extracto sin expediente (Initiative without initiative code), although we could find the following patterns too: “<PAG: Number>” which shows the page number and “*”. Finally, a general XML file is generated which contains all the references to the XML files created in the previous step, organized in legislatures.

7.5 Related work

Other approaches to the PDF to XML conversion have been found in specific literature. In [30, 46, 48] we have found three types of systems for converting PDF documents into XML. In [46], Gurcan et al. describe a converter of articles

of newspapers. In [21], Campos et al. present a system for extracting the text from PDF documents and later indexing this text is described.

The main similarities and differences we can find in our work with respect to the other ones are: We use an intermediate, free converter called *pdftotext* to convert PDF documents into text files and later we work with these files to find the bounds of the XML tags. Finally, the XML files are created. The heuristics used by *pdftotext* (we can see them in the *Xpdf* library[1]) are very similar to the heuristics used by Déjean and Meunier in [30]. Text objects are extracted from the PDF and a word and line segmentation is produced based on heuristics using the distance between characters and their geometrical positions.

In [46], the *PDF Semantic Extractor* starts by analyzing lines and drawings on the page. By the use of a set of customizable filters, only vertical and horizontal lines that can be considered as separators are left. A custom closed-path algorithm is used at this stage, which draws boundaries around zones. Each zone is defined as an area that can contain one or more articles.

For our work, this methodology is not necessary because the PDF documents are not divided in different zones. We only had one problem with these documents because many pages are organized in two columns and it is necessary to read the first column before and the second column later but this problem is solved by *pdftotext*, adding one parameter in the call to the program.

To detect the headers and footers, [30] uses a methodology based on the textual variability since this is much lower in these sections of the page than in the body page. This is a useful method when there are some different types of headers or footers in the same document, but in our case the type of these is always the same, so we only have to look for them and mark them.

The reading order is an important problem in [30, 46], since there could be some articles in a page placed in different zones, in this case we are dealing with multi-column articles. If the reading order is not controlled, we could read parts of different articles in a line. To avoid this problem, [30] uses tables of contents and [46] generates a text block construction methodology. However, we do not have to solve this problem in our case because when we have the text files, the right reading order is from the top to the bottom.

[1] http://www.foolabs.com/xpdf/

The final generated XML files by [30, 46] have a structure focused on the distinct sections whose PDF documents contain, however our XML files have a very defined structure based on the information we require to develop the information retrieval engine later.

The extraction methodology used in our work is very similar to the one used in [21], but the main difference is the chosen tool to extract the text from the PDF documents. [21] uses *PDFbox*[1], a free source library that is able to manipulate PDF documents, but we have seen it has some problems with special characters and the different fonts that a PDF could include. So, we decided to use *pdftotext* which does not present these errors in a MS Windows platform.

The PDF documents used by Hardy and Brailsford in [48] contain the tags to generate the XML files, since from the version 6 of PDF allows a user to create a file containing structural information. However, our documents do not contain such information, because in the corresponding moment it was not considered this possibility.

In [107], Yildiz shows a tool for extracting table information out of PDF files and storing the extracted information in an XML document. To extract the text of the PDF documents, they use the *pdftohtml*[2] extraction tool because it retrieves the positions of the blocks of text, which are very helpful to distinguish the different cells of the table. In our case, we do not need this information due to there are not tables in our PDF documents; therefore, the XML files obtained with this tool contain a group of tags that are not what we use in our XML files, so we prefer the *pdftottext* converter.

7.6 Some information on the XML collection

Table 7.1 shows information about the original collection of PDF documents and the XML collection. All this information is grouped by the legislature and we consider both types of documents: Official bulletins (B), whose DTD is described in fig. 7.8 and its tree structure is shown in fig. 7.9, and Records of parliamentary proceedings which are divided in Plenary sessions (P) and Commissions (C). For

[1] http://www.pdfbox.org
[2] http://pdftohtml.sourceforge.net/

Legislature	Type	#Docs	S-PDF	S-XML
VIII	P	69	64	20
VIII	C	299	183	67
VIII	B	450	603	117
VII	P	141	131	39
VII	C	517	304	98
VII	B	785	1005	148
VI	P	142	133	40
VI	C	412	242	113
VI	B	598	638	145

Table 7.1: Information about digital library.

each period and type of document, we can see the number of documents, the sizes of the PDF and XML collections expressed in MBytes (S-PDF and S-XML, respectively).

Analyzing the sizes of the collections, we can confirm that it is really interesting to store the collections in XML format because its size is much smaller than the collections in PDF format, so it would involve a good size saving.

7.7 Conclusions and future work

In this chapter we have presented the XML digital library of records of parliamentary proceedings from the AP. The main motivation for this conversion has been the possibility of using a structured information retrieval system to allow the access of the relevant elements with a more appropriate granularity of the retrieved items. The internal organization of the documents has been described as well as the conversion process by which, starting from the records of parliamentary proceedings in PDF, they are converted into XML.

Regarding the conversion process, we have to highlight that the applied methodology could be easily adapted to similar types of official documents belonging to different institutions, which contains a well defined structure and, therefore, used to convert them.

With respect to future works, it would be very interesting to get a complete

```
<!ELEMENT BOPA (NUMERO_BOPA, FECHA_BOPA, LEGISLATURA, SUMARIO,
CUERPO)>
<!ELEMENT SUMARIO (SECCION+)>
<!ELEMENT CUERPO (SECCION_CUERPO+)>
<!ELEMENT SECCION (CODIGO_SECCION, TITULO_SECCION, SUBSECCION+)>
<!ELEMENT SUBSECCION (CODIGO_SUBSECCION, TITULO_SUBSECCION,
(SUBSECCION+ | (INICIATIVA | INICIATIVA_MULTIPLE)+))>
<!ELEMENT INICIATIVA (CODIGO_INICIATIVA_SUMARIO,
EXTRACTO_INICIATIVA_SUMARIO?, TRAMITE_SUMARIO?, PAGINA)>
<!ELEMENT INICIATIVA_MULTIPLE ((CODIGO_INICIATIVA_SUMARIO,
  EXTRACTO_INICIATIVA_SUMARIO?, TRAMITE_SUMARIO?)+, PAGINA)>
<!ELEMENT SECCION_CUERPO (CODIGO_SECCION_CUERPO,
TITULO_SECCION_CUERPO, SUBSECCION_CUERPO+)>
<!ELEMENT SUBSECCION_CUERPO (CODIGO_SUBSECCION_CUERPO,
TITULO_SUBSECCION_CUERPO, (SUBSECCION_CUERPO+ |
(DESARROLLO_INICIATIVA | DESARROLLO_INICIATIVA_MULTIPLE)+))>
<!ELEMENT DESARROLLO_INICIATIVA (CODIGO, EXTRACTO_INICIATIVA?,
TRAMITE?, INFORMACION_INICIATIVA*, PARRAFO*)>
<!ELEMENT DESARROLLO_INICIATIVA_MULTIPLE ((CODIGO,
EXTRACTO_INICIATIVA?, TRAMITE?)+, INFORMACION_INICIATIVA*,
PARRAFO*)>
```

Figure 7.8: DTD file of the official bulletins

XML collection with all the official documents from the 1st legislature. This process could be a bit complex because the structure of the documents has changed along the years, besides the format of the documents. So, the methodology could be modified to be adapted for the different documents but minor changes would be done.

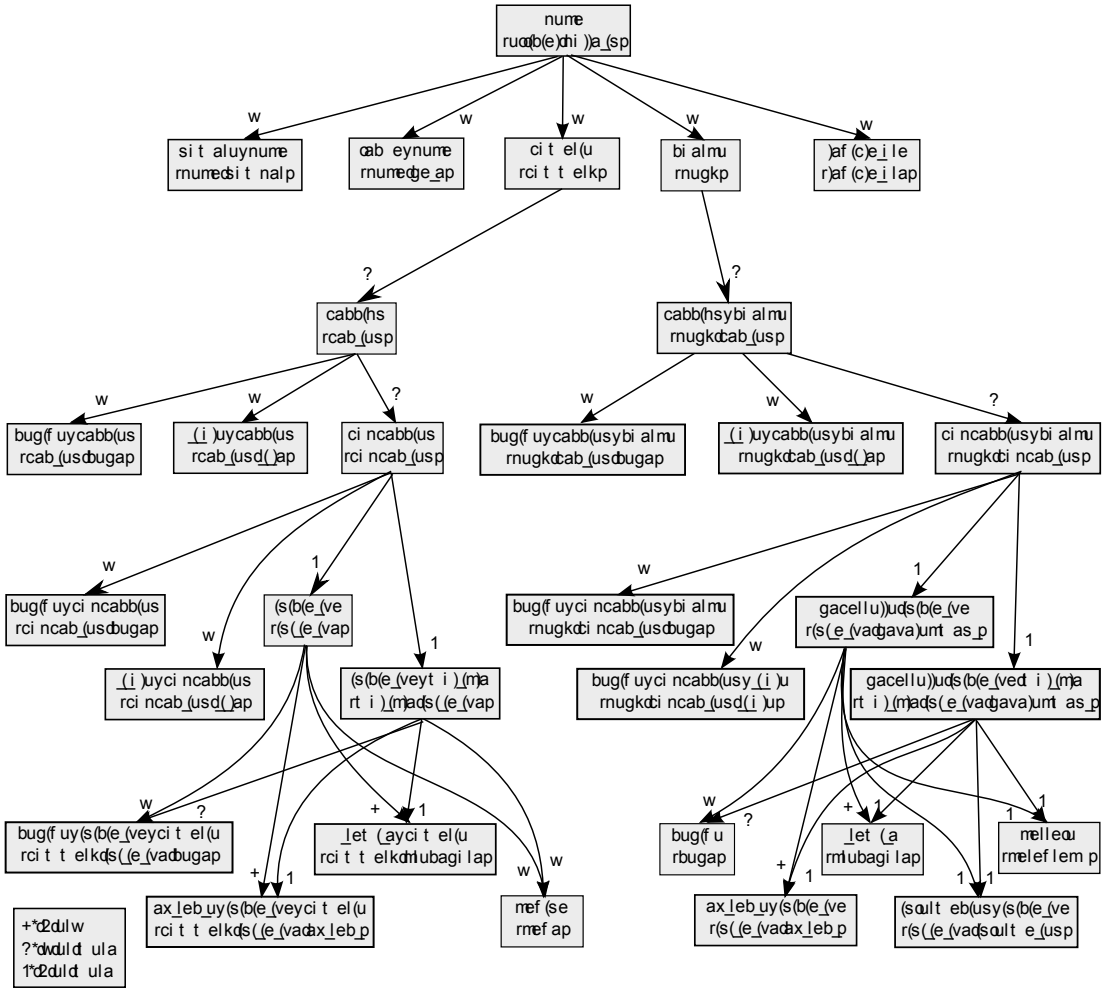


Figure 7.9: Tags structure of a official bulletin.

Chapter 8

Managing videos: Segmentation, synchronization and retrieval

8.1 Introduction

There are some situations in which the information system from a company or an institution has to deal with multimedia information in its daily operation. In some cases, the same information is represented in different but complementary multimedia formats. For example, and focusing on the context of this thesis, the records of parliamentary proceedings contain the exact transcriptions of the speeches of the MP (Members of Parliament) relating to every matter discussed in the different sessions. But also, these sessions are recorded in video.

Therefore, there are two information sources that are implicitly related, as they represent the same event and could be very useful in conjunction. But sometimes this last information, the reference to the event, is the only information able to link them, as due to the fact that their formats are different, there is no possibility of linking them by their content. The consequence is then that the multimedia sources representing the same event are not able to be used in an integrated way. Therefore, new tools that try to synchronize both information sources must be developed.

A common example of this separation in terms of use is a search application for textual documents: users formulate queries in order to obtain relevant documents that satisfy their information needs. Then, once the search engine has returned

the list of documents for a given query, the user has to inspect one by one all of them in order to determine if each document satisfies her/his information need. If the user is interested in watching the associated video, then she/he has to look up the video and access the relevant part inside it. This is a very time consuming task, and by this reason not very useful for the user. The ideal situation would be that the user could read a text and, at the same time, have the possibility of watching the corresponding segment of the video, and with the minimum effort.

A possible solution could be implemented by a (semi-)automatic process where each component of the structured document is associated to its correspondent segment of the video. The way to obtain these structured units has been explained in Chapter 7.

Then, coming back to the search engine context, this application returns the relevant parts of the documents, as well as direct pointers to the related video segments.

Focusing on the Andalusian Parliament (AP), the records of parliamentary proceedings are also recorded in video, so additionally to the transcriptions, the digital library of the Parliament is complemented with the session recordings.

Currently the videos are accessed separately from the text by date and there is no link between the document of the record of parliamentary proceedings and the video. But, it could be very useful for the user that when she/he retrieves a relevant document (the text), or a portion of it, she/he could watch the associated video at the same time. This feature could be an added value for the search engine.

But before the IR System could use the videos to be presented to the user, there is a previous work of synchronizing the text contained in the record of parliamentary proceedings and the corresponding video. The output of this stage would be the XML elements containing text from a speech marked up with time tags, corresponding to the start and end points in the associated video.

From beginning of 2008, the AP is working with a new video management system that is able to support the creation of smaller videos corresponding to each one of the Members of Parliament (MPs) who speak in each session. Although manual, this is a relatively fast process carried out by video operators. Given a complete video of a session and its agenda, the result is a set of XML files (let us call it XML Speech files) containing, for each speaker, basic information about

the legislature, the initiative code, and temporal information about when her/his speech begins and finishes. Moreover, each XML Speech file has got associated a new video containing the speech recording. Then an automatic process of synchronization is performed in order to include the temporal stamps from the time file to the XML file containing the transcription of the session (let us call it XML Transcription file).

But for videos previous to 2008, this new system can not be used so the approach has to be rather different: automatic video segmentation, i.e., the video is automatically partitioned in segments of inner similar content, detecting the boundaries among them, and later synchronized by hand. Therefore, a change of camera is the way to detect the boundary. So, when it happens, a new segment is created. Finally, these segments are associated with the textual transcription of the speeches. Basically, the requirements of this piece of software are:

1. Automatic segmentation of videos.

2. Manual edition of the segmentation.

3. Annotation of the XML documents containing the transcriptions, synchronizing them with the segments of the videos.

Most of the existing segmentation algorithms found in the specific literature are designed for general videos, as Camastra and Vinciarelli describe in [13]. This means that they are complex algorithms prepared to detect the boundaries of the segment in all conditions. But in our context, a simple algorithm based on histogram comparison could work very well. In this chapter, we describe the algorithm itself, how it has been improved, as well as the main features of the segmentation and annotation tools.

Therefore, this chapter describes the two approaches followed for synchronizing the session videos and transcriptions. Then, the chapter is organized as follows:

Section 8.2 will describe a synchronization strategy for transcriptions and videos generated with the new system. The segmentation algorithm, as well as the synchronization method used for older videos, is presented in section 8.3. In

addition, it includes a subsection 8.3.1 presenting a brief review of the available annotation tools. The chapter will end with section 8.4, where some details about the video players are described, and section 8.5, containing some conclusions and further research.

8.2 Synchronization of video and records of parliamentary proceedings via XML files

The first approach that we present in this section is used when, from the current video management system, the video operator generates an XML file per speaker from a session recording (we shall call these files XML Speech files), containing to a great extent the following information:

1. Session identification: date, number of legislature, type and number of session.

2. Name of the speaker.

3. Speech information: start and end times inside the full session video, speech length, name and location of the video file.

The following XML extract is an example of XML Speech file:

```
<dcm>
<track>
<title> Intervencion de "Arenas, Javier" en "Pleno" el 26/06/08.
</title>
<timecode>26/06/08 09:03:15</timecode>
<length>08:05:55.160</length>
<clip_in>00:02:05.000</clip_in>
<clip_out>00:13:31.000</clip_out>
<elements>
<element name="speaker" content="Arenas, Javier"/>
<element name="legislature.ordinal" content="VIII"/>
<element name="session.label" content="PL-VIII/2008-JUN-26/1"/>
<element name="session.type" content="Plenary session"/>
<element name="session.abstract" content="Record of Parliamentary Proceeding,
```

```
no. 13"/>
</elements>
<resourcesCo>
<resource>
<url>file://media/2.wmv</url>
</resource>
</resourcesCo>
</track>
</dcm>
```

This same video operator also generates a shorter video containing only the associated speech to this file.

The synchronization process takes as input the sets of XML Speech files from a session, as well as one XML Transcription file containing the complete session. The result of this stage is the same XML Transcription file completed with a link to the corresponding video as an attribute of the "speech tag". As an example, we can see this fragment of XML Transcription file with pointers to the videos.

```
<development>
<initiative>
<extract> 8-08/DG-000009 . General debate about the reform of the regional
financing.</extract>
<participation>
<speaker> Ms. Fuensanta Coves, President of the Andalusian Parliament </speaker>
<speech videoXML="VIII/DSPAP8013-1/1.flv">
<paragraph> Good morning, ladies and gentlemen. We are going to start the debate
of the initiative 8-08/DB-000009 about the regional financing.
</paragraph>
<paragraph> I gave the floor to Mr. Arenas. Please, whenever you wish.
</paragraph>
</speech>
</participacion>
<participation>
<speaker>Mr. Javier Arenas </speaker>
<speech videoXML="VIII/DSPAP8013-1/2.flv">
<paragraph> May it please the Court.  </paragraph>
<paragraph> Ladies and gentlemen, following with the discussion of the
previous speeches, I would like to remark...</paragraph>
...
</speech>
```

```
</participation>
<participation>
<speaker> Ms. Fuensanta Coves, President of the Andalusian Parliament
</speaker>
<speech videoXML="VIII/DSPAP8013-1/3.flv">
<paragraph> Thank you very much, Mr. Arenas. Now it is the turn for...
</paragraph>
</speech>
</participation>
...
</initiative>
...
</development>
```

Firstly, the matching process consists of considering two different sets of segments or speeches. The first set is composed of the different XML Speech files corresponding to the XML files created by the video operator. On the other hand, the second set is composed of the different speeches which appear in the XML Transcription file, so there must be as many segments as "speech" tags appear in the Transcription file.

In the best case, these two sets should contain the same number of elements, therefore the matching process would be very simple because the system would only have to assign every video from the first set to every "speech" from the Transcription file (second set) in order of appearance in the session, but it is not very usual in most of the cases.

As the video operator works, the speeches of the deputies with a duration less than 5 seconds are discriminated and included in the previous video whose length is more than 5 seconds, so these segments do not have their own XML Speech files. Consequently, in all these cases there are more elements in the second set than in the first one (as we can see in fig. 8.1). Then, we have to distinguish these speeches in the Transcription file too. In order to do that, we have estimated these speeches are those containing less than 25 words (we assume that 25 words $\approx$ 5 sg.), since in the Transcription file we do not have time stamps which indicate the duration of the different speeches.

Due to this estimation could be wrong in some cases, the matching process is a bit more complex. Every video and speech is characterized by the speaker

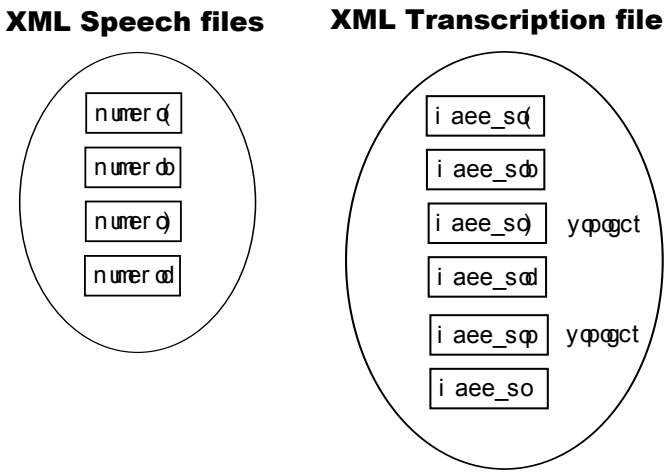


Figure 8.1: Sets for synchronization

of the speech (this information is stored in the XML Speech files and the XML Transcription file), then the matching process compares the names of the speakers of the two elements before assigning the element from one set to the other one. In case they are equal, the assignation is done, but in the opposite case, the algorithm takes the following element from the second set and compares again.

We can see an example of the synchronization process in fig. 8.2. In videos 1 and 2, the matching process has been very fast because the speakers of the two first speeches of the XML Transcription file correspond to the speakers of both videos. However, the speech 3 has been discarded because it contains less than 25 words (less than 5 seconds). Then, the speaker of the video 3 corresponds to the speaker of the speech 4. Afterwards, the speech 5 is discarded and the speaker of the video 4 corresponds to the speaker of the speech 7, since the speech 6 has been discarded too, because the speakers are different.

The assignation process is based on including the needed data in the XML Transcription file, basically an attribute containing the path and the name of the video: $<speech\ videoXML = \text{“}VIII/DSPAP8013 - 1/1.flv\text{”}>$.

Finally, the XML Transcription files, as well as all the speech videos, are provided to the IR System so it could index them, first, and retrieve them, later.

As it may be realized, the video formats are different in the XML Speech video and in the XML Transcription video. The reason is that for visualization

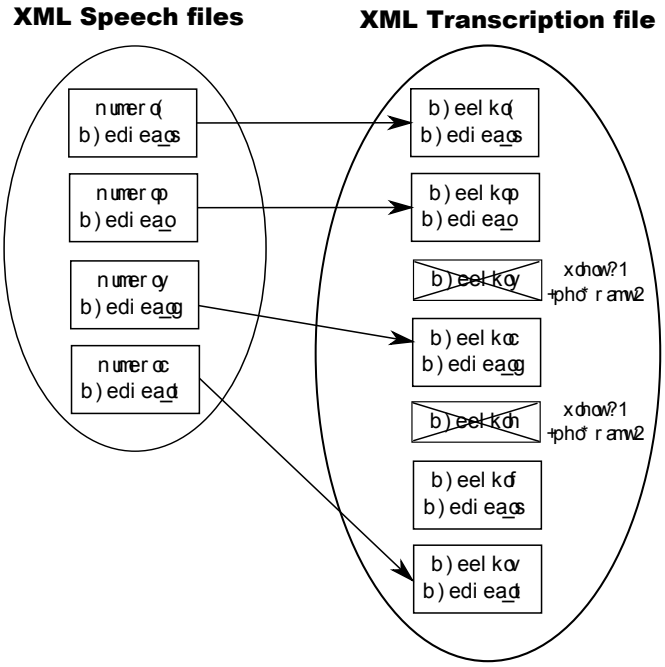


Figure 8.2: Example of synchronization

reasons, explained in section 8.4, the IR System will show a Flash video instead of the Windows Media Video from the original recording.

8.3 Synchronization of video and records of parliamentary proceedings via video segmentation and annotation

8.3.1 Revision of existing annotation tools

Having a look at the available tools for video annotation, we could find, among others, the following ones:

- *IBM MPEG-7 Annotation Tool*[1]: it allows annotating videos using the metadata associated to a MPEG-7 video. This is segmented automatically, and a key frame extracted. On this representative, the user can make the

[1]http://www.alphaworks.ibm.com/tech/videoannex

pertinent annotations about events of static scenes and key objects. For our purposes, the annotation is very restrictive, because it only can be done using predefined scenes and objects. Also, it is in MPEG-7, which is effectively a standard, but we are not considering this multimedia format. Another problem is that the segmentation can not be edited.

- *IBM Multimedia Annotation Tool*[1] is very similar to the previous one, but also can annotate audio, background and foreground sounds. Both tools are free.

- *Virage VideoLogger*[2] analyzes the video and generates automatically a structured index about the content. It allows making manual annotations, but it is not free. This tool is really complex because of its high functionality.

- *iFinder*[3] is another tool which works with the MPEG-7 standard. It is able to segment videos, recognize faces, as well as to make speech recognition. It is a client/server application. But, it is not free and really complex.

- *Ricoh MovieTool*[4] allows automatically segmenting MPEG-7 videos, but giving the possibility of refining the output, making it manually. A very interesting feature is the possibility of creating a hierarchy of segments.

- *ZGDV VideTo*[5] presents the interesting feature that the annotation could be stored in XML format, without any restriction (later it could be converted to MPEG-7). Patterns about specific domains could be created, facilitating the annotation. The segmentation is fixed and can not be edited.

With the analysis of all these tools, we realize that there is no tool that fulfills all our requirements, so a new one is necessary for our purposes.

[1] www.alphaworks.ibm.com/tech/multimodalannotation
[2] www.virage.com
[3] www.imk.fhg.de/de/ifinder
[4] www.ricoh.co.jp/src/multimedia/MovieTool/about/index.html
[5] www.zgdv.de/zgdv/departments/zr4/Produkte/VIDETO

8.3.2 Parliamentary solution

In this section the solution provided for those older videos not managed by the new video management system is described. In this case, the solution could be closer to the philosophy underlying in all the tools presented in Section 8.3.1, but as the features of the problem are very different, it is also more appropriate to find a solution that fits well with the AP's context.

In this case, the input is the full video of the session, and the output is the XML Transcription file annotated with start and end stamps in the desired tags, with respect to that original video. Moreover, the synchronization could be made not only at speech level, but also to paragraph level. The general process of this second approach is as follows: firstly, the video is automatically segmented in pieces (each segment will contain a high internal homogeneity); secondly, the segmentation could be manually modified to solve any problem, or to create a more refined segmentation; and finally, the XML Transcription file is manually synchronized with respect to the previously created segments.

When the annotation of a video is carried out, we have to present the information to the user in such a way that all the content is summarized in a small amount of information. An adequate way is to split the video in segments [13]. A segment is a fraction of video which has continuity. Once the video has been divided, it must be presented to the user, in such a way that with only a look she/he could have an idea of the content of each segment. This is made showing a keyframe, a representative of the content.

The segmentation, based on difference between shots [52, 47, 80], obtains some features from each shot and compares them with those from the following shot. If the difference is greater enough, then it is considered that there is a change of scene. The most used features are the colours [44] or the edges of the image [41]. To obtain the differences of the shots, the format of the video storage could consider [11] shots storing the full image, or those storing only the change with respect to the previous shot. The segmentation based on flow of movement tries to detect the object in the scene (foreground and background objects) and analyzes the movements in the following shots [59, 4, 63]. There also exist hybrid techniques which combine both methods [62].

The objective is to find the method that is able to segment the videos of the AP, in a correct and efficient way, without forgetting the complexity of the implementation, which has to be low. These videos show long scenes, with few movements, and sudden changes. This means that the analysis of objects in the scene is not going to help, because the speakers are usually static in the talk.

In the chamber of the Parliament, there are four fixed cameras, focussing to the speaker, a general view of the chamber, and two centred in the seats of the MPs. Therefore, the realization of these videos is usually very static, and therefore very easy to detect the changes of cameras. Considering the segmentation using shot differences, we shall notice that in the changes of camera, these differences are large, and inside the same segment, the differences are low, because the cameras are static and the movement is almost null. Therefore, we shall use this last method, considering the colour as the feature in which the method will be based on, basically because of the simplicity of the method itself and good results.

8.3.3 The segmentation algorithm

The segmentation algorithm that we present in this section is based on detecting differences between shots. More specifically, these differences will be given by the different colours of the shots. We shall adopt a gray scale representation.

From the RGB images of the video, we get that representation using a simple transformation:

$$I(i,j) = \frac{(R(i,j)+G(i,j)+B(i,j))}{3},$$

where I is the matrix that represents the image in grey scale; R, G and B are the matrices that represent the levels of red, green and blue of the image, respectively, and i and j are the indexes of a pixel.

The histogram of the image represents the number of times that a specific colour occurs in it. As we are working in gray scale, and considering that 8 bits are used to represent the intensity of 256 tones, we could represent the image by means of vectors of 256 elements. If we note H the vector, and $H(i)$ the number of occurrences of the colour i in the image, then, in order to determine if there has been a change between two shots S_1 and S_2, with histograms H_1 and H_2, respectively, we could compute the difference of both vectors:

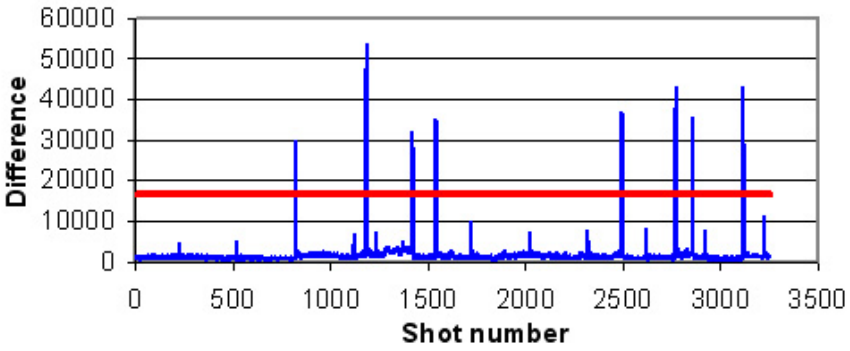


Figure 8.3: Differences between shot histograms

$$(H_1 - H_2)[i] = |H_1[i] - H_2[i]|\,.$$

Computing the difference for each colour, and summing up all of them, we have a scalar value of the difference between both. If this value is greater than a certain threshold, then there is a change in the shots, as may be observed in fig. 8.3. The blue lines are the differences between shots and the red line is the threshold.

This is a really simple algorithm as a set of shots are considered included in the same segment if the difference between their histograms is low. But experimenting with the videos of parliamentary sessions, we realized that the changes of cameras cause that the difference of histograms is high, so they could easily be detected. But, it is also high (although a little bit smaller) for shots in which there is no such change in the camera. Therefore, mistakes might be made. This is due to a minimum variation of the luminosity of the image, which considerably alters the histogram of the image.

For example, a simple movement of an MP who is speaking could provoke that his/her jacket reflects the light in a different way. This could make that the colour represented in a shot is lightly different to the previous shot. Although the difference is not noticeable for the human eye, a minimal variation in the histograms increases the difference between both of them.

In order to solve this problem, we have to achieve that between close histograms the difference is not so important. So, we could soften the histogram, in

such a way that each element depends on the neighbours, and therefore a light change does not affect the difference.

A solution is the application of a convolution filter, which makes that each element of the vector is the sum of those closest elements. We have carried out such convolution using the vector [0.1, 0.2, 0.4, 0.2, 0.1]:

$$H_1'[i] = 0.1 * H_1[i-2] + 0.2 * H_1[i-1] + 0.4 * H_1[i] + 0.2 * H_1[i+1] + 0.1 * H_1[i+2].$$

An important decision that will clearly have a great influence in the performance of the segmentation is the selection of the threshold value. It will depend on the resolution of the video, as in shots with a higher resolution the difference of their histograms will be proportionally larger. Moreover, images with a higher number of colours will also present a higher difference among shots, so we shall have to consider the number of tones contained in the images. Therefore, the threshold used in our algorithm takes into account the width of the image, its height, as well as the number of colours. It is defined as:

$$Threshold = \frac{Width * Height * N^o.\ of\ colours}{Factor},$$

where $Factor$ is a parameter decisive to get an optimal segmentation.

8.3.4 Optimization of the algorithm efficiency

The first parameter that must be estimated is $Factor$, used to compute the threshold just mentioned. For the type of considered videos, its value has been obtained empirically studying several of them.

The process which has led to get it has been the following: firstly, we have obtained the difference for each pair of contiguous shots in each video; secondly, we have localized manually the cuts (changes of scene) that the segmentation algorithm should detect; and finally, once we have studied the values of differences in the shots in which there is a cut, we have selected a value for Factor such as the threshold value is sufficiently low to detect all the real cuts, and sufficiently high to not detect cuts which do not exist. With a threshold of 16,000, all the cuts are detected, and nothing except real cuts will be detected.

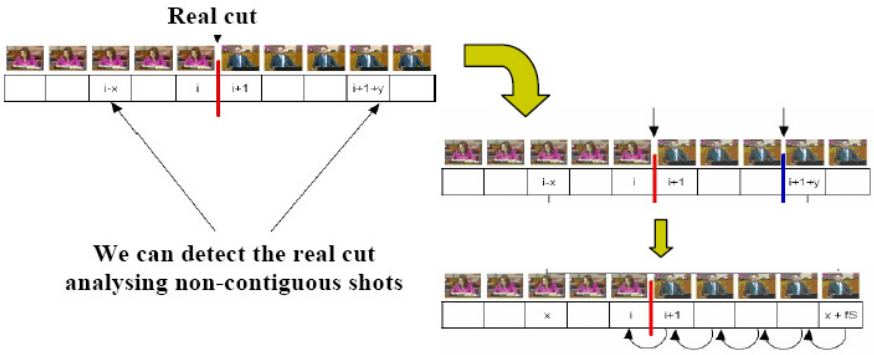


Figure 8.4: Comparison of non-contiguous shots and posterior refinement

This basic segmentation algorithm works properly, but not efficiently. As the videos from the sessions of the AP are very long (about 5 hours), it is required to improve the segmentation speed, but without worsening the effectiveness.

The first attempt is to reduce the number of shots to be considered. In the original algorithm, the histograms of each pair of shots are compared. But it is noticed that if there is a cut between the shots i and $i+1$, this cut also will be placed between the shots $i-x$ and $i+y+1$. This means that we could detect the cut analyzing the difference of histograms between two non-contiguous shots surrounding shots i and $i+1$. Therefore, instead of analyzing all of them, we shall discard n shots between each studied pair.

The following step will be to refine the segmentation to locate exactly when the cut is produced, i.e. if the algorithm finds a cut between shots a and b ($a < b$ and $b = a + n$), then it will visit each pair of shots i and $i+1$, with $a < i < b$, contained in the interval, to detect the cut. This process is much faster than to compare each single pair of shots and also offers the optimal result. For videos in which there is not an excessive number of cuts, as the case is, this is an appropriate method. Fig. 8.4 shows a graphical example of this process.

A second optimization is related to the size of the image. If the difference of histograms with the full image is enough to differentiate shots, we could suppose that the histogram of only a portion of the image could offer enough information to perform this action, adjusting the threshold to the corresponding size. Then the reduction will improve the efficiency of the process, as the number of com-

Figure 8.5: Shot with a reduction factor of 3. The lower left part is considered

putations is lower. This is not only interesting in terms of efficiency but also in terms of efficacy of the segmentation.

In most of the videos, there are many movements inside a scene which do not imply a change of shot, for instance, if we watch a scene of a man moving his arms, all the images from this scene belong to the same shot. Therefore, the algorithm could make a mistake because the movement originates a change in the colours of the image. In the case of the videos of the AP, where the location of the cameras is known, we could know the part of the image that will suffer less interferences of this type. If we only use the pixels of the area to compute the histograms, we reduce the risk of finding cuts where there are not. In the case of study, the lower left corner of the image is considered, as empirically it is the area where fewer movements are observed. Fig. 8.5 presents a shot of the chamber with a reduction factor of 3.

Finally, once the segmentation is finished, we have to select a representative shot of each segment (key frame), in order to present them to the user. There are several techniques to select the key frame [6, 12, 10]. But in the case of the videos of the AP, and due to their nature, it is not required a sophisticated technique as the user is going to use it as a guide to the annotation phase. Then,

Videos	Accuracy with refinement	Accuracy without
V5	100.00	27.78
V9	100.00	87.50
V20	92.59	20.00
V30	88.89	25.53
V40	100.00	76.32

Table 8.1: Percentage of correct segments detected (accuracy) with and without refinement.

the shot which is in the 10% of the total length of the segment is selected. With this selection there are no mistakes, as the segment is supposed to be accurate in terms of very similar contents of all its shots.

8.3.5 Experimentation and evaluation

In this section, we shall show how the selection of the best values for the two main parameters of the segmentation algorithm has been made, so the best performance is reached. In this line, we have to decide the size of the window used to compute the histograms and later the difference between them (reduction factor), and the number of shots ignored in each stage of histogram comparison, and the posterior refinement to find the exact point of change.

We have created a set of 6 videos of different length. Their durations as well as the number of exact segments, obtained manually when there is a change in the scene are the followings:

V5(5m, 5), V9(9m, 7), V20(20m 26s, 27), V30(30m 13s, 36), V40(40m 8s, 29), VF(5h 46m 36s, 433).

The first experiment tries to show how important is to use a reduction factor. In table 8.1 we could show the accuracy of the segmentation considering the first 5 videos, with and without refinement (reduction factor of 2 and 24 shots between two compared shots). As it may be noticed, the refinement is configured as an essential technique to maintain a high accuracy.

With respect to the reduction factor, we have tested that same set of videos with several values, from 1 to 9. In fig. 8.6, we can observe the graphical representation of the reduction factor and segmentation time. Fig. 8.7 shows the

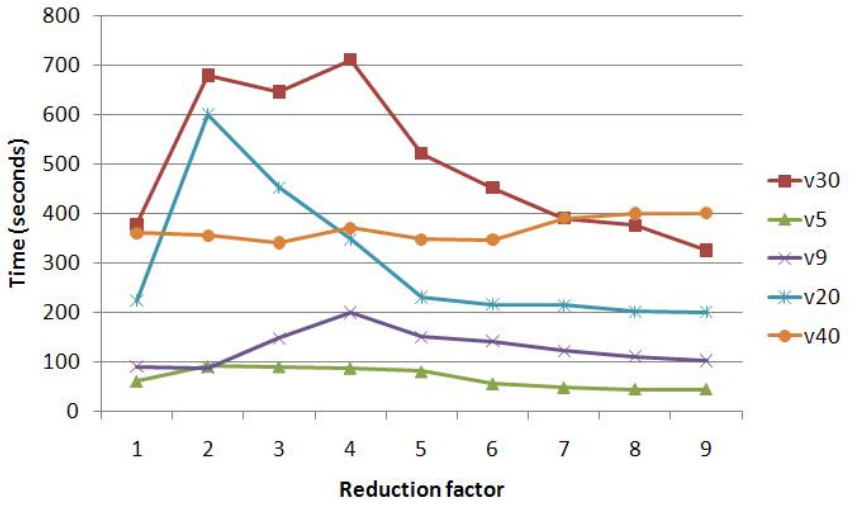


Figure 8.6: Time vs. Reduction Factor

reduction factor and the percentage of correct segments. Initially, these experiments have been carried out with the number of shots jumped set to 24 and refinement applied.

The segmentation time is the sum of times of different tasks: the shot extraction process from the video, the computation of its histogram, the convolution, the comparison with the previous shot, and the refinement, in case of needing it.

In affirmative case, the time would increase considerably, as the algorithm has to look for the change from the previous shot until the shot where the real change is produced. When the reduction factor is increased, the time required to compute the histogram is reduced. But, as a smaller portion of the image is used, there is less information available and therefore there will be more segmentation errors.

These two situations are found in videos V30 and V40. In the former, as there are lots of cuts, the reduction factors, 2 to 7 make the algorithm to spend more time testing the differences between histograms when refining than the time saved using smaller image areas. In the latter, the cuts are less frequent, so better times are obtained with factors 2 to 6.

With respect to the accuracy of the algorithm, we may notice that, in most

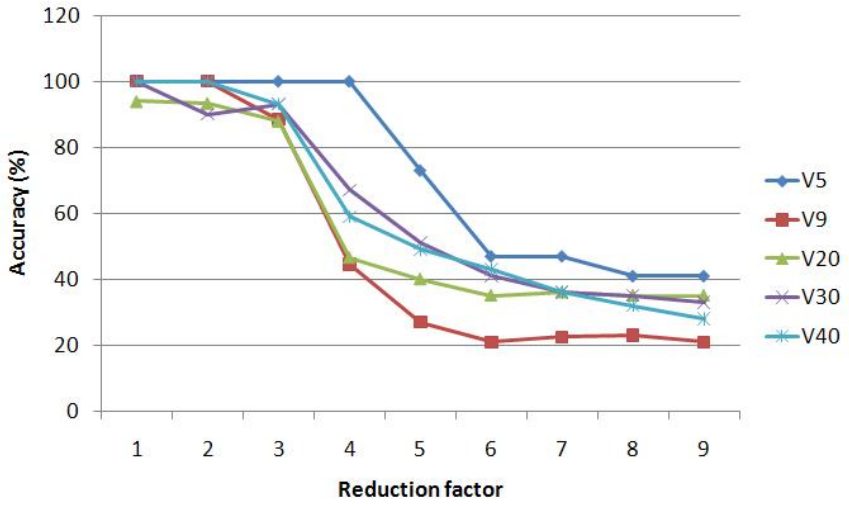


Figure 8.7: Accuracy vs. Time

of the cases, from a reduction factor greater than 2, it decreases considerably. This is due to the fact that there is less information about the change of images testing a small area, so the algorithm makes more mistakes. As a conclusion, we shall use a reduction factor of 2 because the accuracy is usually very close to the 100%, and the segmentation time is usually one of the shortest.

Fixed the value of this parameter, next we shall study the number of shots ignored between two shots to be compared. In this case, the analysis will be done with VF in two steps. Fig. 8.8 shows the segmentation time for values 1, 12, 22, 32. We can observe how the value 12 gets a better performance. In a second step, we have tried with values 10, 12, 14, 16 and 18 (see fig. 8.9). In this case, 14 is the best.

8.3.6 Features of the segmentation application

Before starting with the main features of the segmentation tool, just to mention that it has been developed in Java, using the Java Media Framework (JMF) library to incorporate multimedia elements and tasks in our application in a comfortable and easy way.

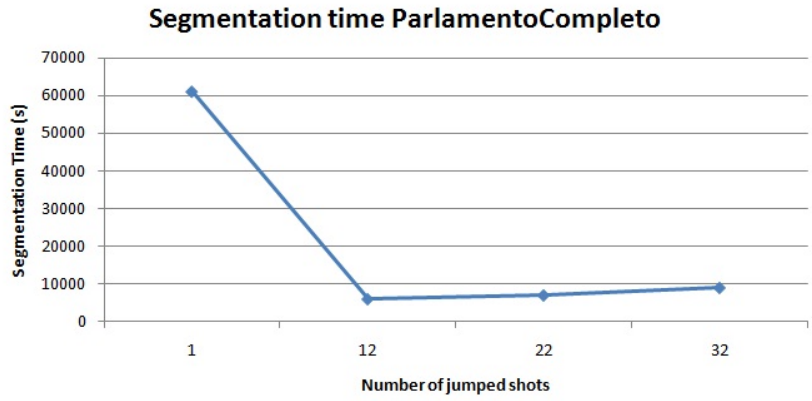


Figure 8.8: Segmentation time for different numbers of shots ignored (first step)

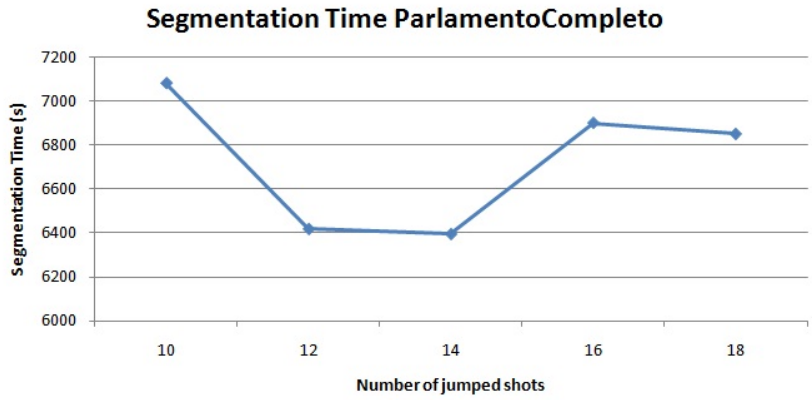


Figure 8.9: Segmentation time for different numbers of shots ignored (second step)

There exists the possibility of making a batch segmentation of several videos. This facility is implemented because the segmentation time is usually long taking into account the average duration of them. Then, the user selects a group of videos to be segmented, and the application will process each one, given as an output the segments of each video.

Once the automatic segmentation of a video has been carried out, the software offers the possibility of editing the segmentation manually. The output of this process is a set of segments, represented by a key shot. The user may need to adjust the segmentation to prepare the posterior process of annotation, in order to be more accurate.

An example of this need is the case in which a deputy is speaking, and a camera focusing her/his. Then, there is a change of camera, and that focuses another deputy. In spite of this change of scene, the first speaker is continuing with her/his talk.

The segmentation algorithm would give as an output two segments (correct), but the user may wish to join both of them, because the current speech is being given in both of them. Therefore, the user is allowed to edit the segments, combining them (two segments s1 and s2 could be joined if they are contiguous) or dividing segments in two.

In the application, all the segments found by the algorithm are shown in a window (see fig. 8.10). More specifically, the key shot of each segment. If we click in one of them, then all the shots contained in it are shown in a separate window. By means of submenus activated by the left bottom of the mouse, the user could edit the segments. A viewer allows playing any segment.

In order to show a shot, the application must access the video. This means a high access, the player must be placed in the correct position, the file in the hard drive must be accessed and copy the contents in main memory. For a real time usage, this would mean that there is a high response time. To avoid this, we have implemented a cache module, in which we store the shots of the video in a cache memory (those which are more probable to be shown), so their access will be much faster.

Figure 8.10: Screen shoot of the Segmentation tool

8.3.7 The synchronization tool

Once the segmentation of a video has been performed, and the posterior edition, the user is ready to carry out the annotation stage. The input of this process will be the sets of segments found in the video corresponding to a parliamentary session and the transcription of the speeches given in the chamber for that video. This transcription is represented by means of an XML document, which contains the structure of the session, as well as the text itself. The output will be the XML document containing the transcription synchronized with the video by means of time stamps in the elements of the document. The segment of the video related to a specific text could be easily accessed.

The annotation tool (fig. 8.11) will consist of a manual association of segments with the corresponding elements in the XML document. So, each tag will have a link to the part of the video where this text is played.

Fig. 8.11 shows the user interface of the annotation tool. It is composed of four windows. The left window shows the tree representation of an XML document containing a record of parliamentary proceedings. If a leaf node is clicked, then the text contained in it is shown in the central uppest windows. The window below contains the segments found in the first part of the process.

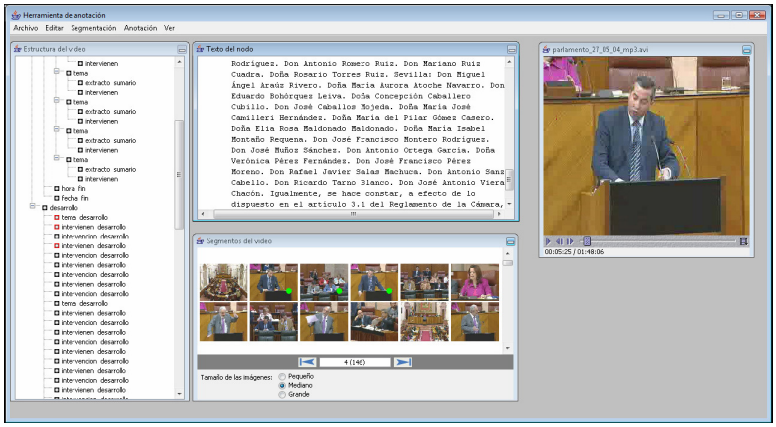


Figure 8.11: Screen shoot of the Annotation tool

Finally, a player is included in the right part of the interface, in order to help the annotation.

The annotation process is as follows: the user selects a segment of video from the window below, then looks for the node in the XML document from the left window containing the transcription of the audio of that segment, and by means of a drag and drop action, associates the former with the latter. These steps are repeated until all the segments have been assigned to a node of the document.

Actually, with the association of a segment to an XML element of the document, we introduce a pair of attributes to the desired tag, containing the start and end times of the segment (for example, *<speech bt="00:45:54", et="00:59:02">*), plus a video tag pointing to the full video of the session at the beginning of the XML Transcription file. This information will be enough to access the portion of the video in retrieval time.

Once all the segments have been assigned to leaf nodes of the XML tree, and therefore, all the affected tags have been complemented with temporal attributes linking the text with the video, it is necessary to propagate the times to upper nodes until reaching the root node. In our context of the parliamentary sessions, the text is usually contained in 'paragraph' tags. These are contained in 'speech' tags (the speech of a deputy), which are included in the debate of a parliamentary initiative (several MPs usually takes part in the discussion of a point of the agenda).

Finally, several points of the agenda, plus general information of the session, are included in the 'session' tag, the root of the tree. Then if the search engine retrieves a whole point of the agenda, it should also allow the access to all the segments related to the elements contained in that tag. Therefore, a propagation of times is performed from leaves to the root node, where the start and end times of the container units correspond to the smallest start time and the biggest end time of the direct descendant units, with timestamps, of them.

8.4 The video player

Once the transcriptions in form of XML documents have been completed with references to the video (first approach) or time stamps (second approach) and the parliamentary digital library has been indexed, the search engine is ready to be used. When a user query is formulated to the system, it ranks all the elements from all the documents according to their relevance with respect to the query, and presents this ranking to the user. Then she/he could click on a link, in which she/he is interested in, to inspect the text or to watch the video segment, which is played by means of a player (fig. 8.12).

An important element in the retrieval stage is the video player. We have to take into account two important features in order to make the decision of what type of player to use: first of all, it must be an accessible element for any type of user, independent on the platform used; secondly, in order to save broad band in the video transmission, we only have to send the portion of the video in which the user is interested in (not the whole video) to the player from the server. Usually, the videos we are working with are very heavy, so the user can not be waiting to download 40 Mbytes of video to watch only 30 seconds. This must be fast.

For this reason, the player is implemented using the Flash technology. Adobe Flash is a multimedia platform used to add animation, video, and interactivity to web pages. It manipulates vector and raster graphics to provide animation of text, drawings, and still images. It supports bidirectional streaming of audio and video, and it can capture user input via mouse, keyboard, microphone, and camera. Moreover, Flash implements a programming language called ActionScript. One

Figure 8.12: Screen shoot of the search result screen with the video player

of the main features of this language is to incorporate a multimedia player inside a Flash animation by means of its libraries.

An important additional feature, by which we made the decision of using this technology to develop the player used in the search engine for the digital library of the AP, is the possibility of playing a piece of a video in any time of it, without downloading the whole video in the computer.

8.5 Conclusions and future work

In this chapter we have presented two approaches to tackle the problem of synchronizing video and text, corresponding to the recordings and transcriptions of parliamentary sessions. The videos record the sessions, while the documents, represented in XML, store the exact transcriptions of the speeches given in the session. From the user point of view, it is very interesting that, as the output of a query, the system returns both the text associated and its corresponding portion of video.

The first approach takes into account the fact that with the new video management system, the videos corresponding to a complete session are divided in so

many shorter videos as speeches. Then with the XML Transcription file and all the XML Speech files a synchronization process is performed. The output is the former file completed with the name of the files related to each speech.

The second approach relies on a video segmentation and annotation tool for the videos and the documents. To implement this, we need a first step in which the video is segmented in pieces, and later, each interval is assigned to a text element belonging to the XML document. Therefore, in this chapter, we have presented a very simple but effective and efficient segmentation algorithm to carry out this task, as well as an annotation tool to synchronize both media. Both tools have been designed thinking in the easiness of use.

With respect to further research, we think that the human work in the process of synchronization is too much, so we would like to reduce it, changing the philosophy of the tool: instead of segmenting video and later manually annotate it, we shall research on a process in which automatically the audio of the session would be synchronizing with the text of the transcriptions. We think this would improve the process, alleviating the human intervention.

Chapter 9

A web interface for the Andalusian Parliament digital library

9.1 Introduction

In the creation of the IR system for the AP, *Seda*[1], the last step is the development of a web application which follows the client/server paradigm, where the user can interact with the system introducing queries and viewing the results for these queries in different formats. Its user interface has been carefully designed in collaboration with staff at the Andalusian Parliament. The objective was to develop a simple and intuitive interface in HTML (Hyper Text Markup Language) using PHP (Hypertext Preprocessor) as server-side HTML embedded scripting language and CSS (Cascading Style Sheets) style sheets.

Once a user has entered in *Seda*, she/he can access information in the Andalusian Parliament's legislative collection by means of one of the two available forms on the web page. The first one is for CO queries described in section 9.2.1 and the second one is for CAS queries, described in section 9.2.2. After the search engine, Garnata, has generated the results for a query, they are presented to the user following the steps shown in section 9.3. Then, these results can be judged

[1]Stored in the URL: `http://irutai.ugr.es/BuscadorParlamentoA`

by the user with the objective of using this information in the RF framework to generate a new expanded query more adapted to the user's needs; therefore, the interfaces developed for these purposes are shown in section 9.4. Lastly, section 9.5 contains some conclusions and future research.

9.2 Formulating queries

As our collections are composed of structured documents, the interface and the whole interaction must enable the user to make full use of this document structure. Focussing on the query formulation stage, there are several approaches for expressing an information need to the structured IR system:

1. The classical IR approach of providing a set of keywords, known as CO query.

2. The use of a query language specifically designed for querying structured documents.

3. The use of a graphic user interface for formulating CAS queries.

Then, with the first approach presented above, and from a point of view of a user interacting with a structured IR system, the structural restrictions in natural language are very difficult to be captured by the system. So, the user is not getting the most of the document structure. In the second approach, a well defined query language like NEXI [97] allows the creation of CAS queries as we commented in section 4.2 of chapter 4.

The main problems of such languages are two: firstly, the user has to learn a relatively complex language, and later use it properly, and secondly, the structure of the collection has to be very well known by the user to take advantage of both the power of the query language and the collection itself.

Finally, in the third option, by means of graphic components in a form, the system supports the formulation of a query. The two advantages are that a high knowledge of the structure of the collection is not required and a CAS query could be easily formulated. Also, the possibility of making a mistake in the formulation of the query is almost null because the process is controlled by the user interface.

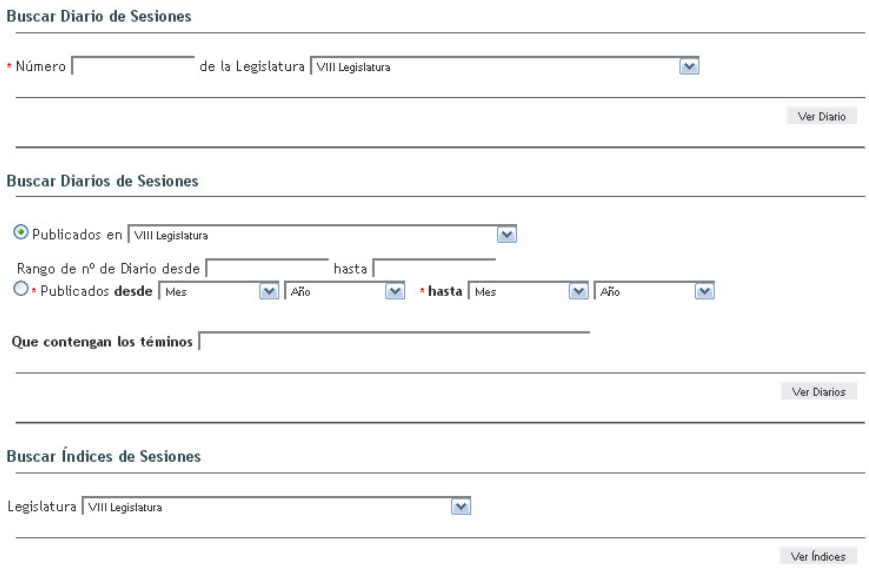


Figure 9.1: Form to introduce queries in the Andalusian Parliament webpage

For expert users, the 2nd option perhaps is the most appropriate, but for common users, the 3rd option is the most suitable.

9.2.1 Description of the user interface for Content-only queries

The form to introduce CO queries in *Seda* is based on the form of the search engine of the AP, which is located in its webpage[1] (see fig. 9.1), since we have tried to keep the same search parameters, but improving the way to access the documents.

In addition to introducing the keyword query in natural language, we can also add several restrictions to it (see fig. 9.2):

- Legislature (*Legislatura*).
- Type(s) of documents where the user wants to search (*Tipo de documento*): This parameter does not appear in the form of the AP search engine, since every search is only for one type of document.

[1] http://www.parlamentodeandalucia.es

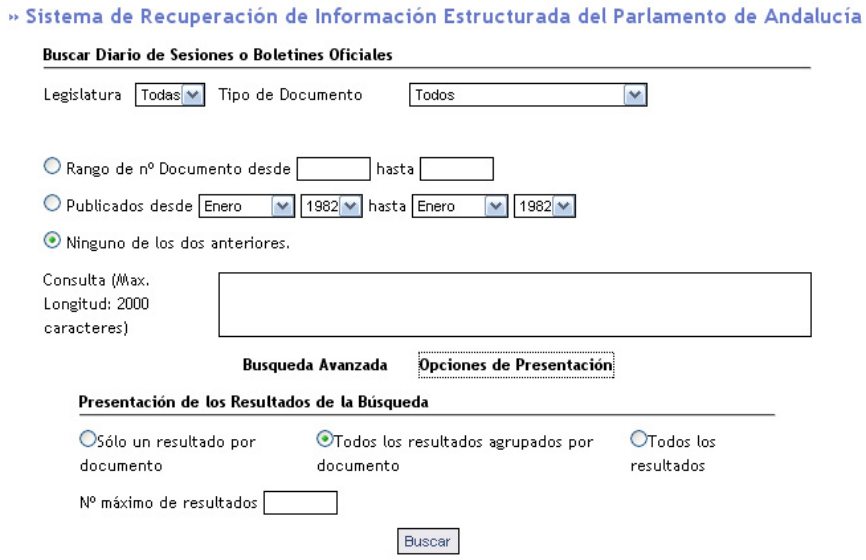


Figure 9.2: Form to introduce Content-only queries in our search engine, *Seda*

- In case of a comission, serie and type of comission.
- Range of document numbers where to search (*Rango de nº de documento desde ... hasta ...*).
- Range of publishing dates of the documents (*Publicados desde ... hasta ...*).

Afterwards, different presentation options are shown to specify how the results are arranged. So, the application helps the users find the units satisfying their information requirements:

- Only one result per document (*Sólo un resultado por documento*): the system will show only one result per document. This single document part should correspond to the best entry point for starting to read the relevant text in the document (corresponding to best in context task of INEX).
- All the results grouped by document (*Todos los resultados agrupados por documento*): for each document, the search engine will return every relevant unit according to its relevance (corresponding to relevant in context task of INEX).

- All the results (*Todos los resultados*): all the relevant units (without any association, after removing overlapping units) are presented to the user in decreasing order of relevance (corresponding to focused task of INEX).

Lastly, we can decide the maximum number of results returned by the search engine (*Nº máximo de resultados*).

9.2.2 Description of the user interface for Content and Structure queries

In this section, the user interface for supporting the formulation of CAS queries is presented. The final output of the query formulation process, and totally transparent to the user, will be a CAS query formulated in the NEXI language, which will be the input to the underlying IR system able to process NEXI queries.

Therefore, and considering the components of this NEXI query, we have to design an appropriate visual method to express a target element (the document element in which the user is interested for retrieval purposes) and its associated text query, as well as the context element(s) (the document element(s) that establishes a restriction over the target elements) and its (their) corresponding text query(ies).

For this purpose, the form is composed of two groups of graphic components: those used to input the information related to the target, and those for the context. More specifically, in each group, the user will find a list box, where she/he could select a unit, plus its associated text field, from where the query terms are introduced in it. In both cases, for the target and context, the list boxes will contain comprehensible labels of the XML tags, instead of their names in the documents themselves, so they are totally transparent to the users.

Specifically, for the target list box, only those retrievable tags are shown, while for the context, only those tags where restrictions can be established, are included. Leaving the text field blank and no element selected from the list box from the context group, the NEXI query will be only composed of the target part ($//C[about(., text1)]$).

In fig. 9.3, we may see a screen shot of the visual query formulation interface (in Spanish), according to the requirements given in the previous paragraphs.

The two differentiated parts, the target (*Objetivo*) and context groups (*Dar preferencia a los elementos en el contexto*), are represented. In the former, the text field (*Texto a buscar*) and the list box (*Etiqueta*) are used to select the textual query and the type of retrievable element.

In the case of the example of the figure, the user is pointing out that she/he is interested in a paragraph of the speech (*Intervención (párrafo)*) dealing with "profesional training" (*formación profesional*). In the latter, following the same philosophy, the user is able to input the context of the search, i.e., restrictions imposed by the selection of other types of elements and the formulation of the associated query.

In the example, the restrictions for those paragraphs of the speech are that preferably contained in speeches where the speaker (*Interviniente*) is the President of the Andalusian Government (*presidente de la Junta de Andalucía*) and the initiative extract (*Extracto de la iniciativa*) is related to the "education law" (*ley de educación*).

The user could include the number of restrictions that she/he considers (using the button with the text "Add restriction" (*Añadir restricción*) and, once formulated, she/he could remove any of them (clicking in the corresponding black cross on the right hand side). If the user selects the special label in the list box of the target group named "Any" (*Cualquiera*), she/he is asking the structured IR system to return any type of relevant element.

As mentioned before, this design has been implemented in the web application for the Andalusian Parliament [27]. Then, the user can introduce queries, but every query is only for one type of documents (records of parliamentary proceedings or official bulletins) because most of the tags of both types of documents are different being impossible to create a CAS query combining units from both types of documents. The different tags included in the list boxes of the context and target parts for the two types of documents are:

- Record of parliamentary proceedings:
  - Context:
    - Deputy who participates in the session (Interviniente).

        * Debate of the complete initiative (Debate de iniciativa completo).
        * Complete speech (Intervención completa).
        * Paragraph of the speech (Intervención (párrafo)).
        * Initiative extract (Extracto de la iniciativa).
        * Legislature (Legislatura).
        * Session number (Número de diario).
        * Serie (Serie).
        * Type of session (Tipo de sesión).
        * Plenary session number (Número de sesión plenaria).
        * Day of the week of the session (Día de la semana de la sesión).
        * Year of the session (Año de la sesión).
        * Month of the session (Mes de la sesión).
        * Day of the session (Día de la sesión).
        * Matter (Materia).
    - Target:
        * Paragraph of the speech (Intervención (parrafo)).
        * Debate of the complete initiative (Debate de iniciativa completo).
        * Complete speech (Intervención completa).
        * Initiative extract (Extracto de la iniciativa).
        * All tags (Cualquiera).

- Official bulletin:
    - Context:
        * Official bulletin number (Número de BOPA).
        * Paragraph of the initative (Iniciativa (párrafo)).
        * Initiative extract (Extracto de la iniciativa).
        * General information about the initiative (Información general sobre la iniciativa).
        * Procedure (Trámite).

    - Initiative code (Número de expediente).
    - Official bulletin date (Fecha del boletín).
    - Legislature (Legislatura).
  - Target:
    - Initiative extract (Extracto de la iniciativa).
    - Paragraph of the initative (Iniciativa (párrafo)).
    - General information about the initiative (Información general sobre la iniciativa).
    - Procedure (Trámite).
    - Initiative code (Número de expediente).
    - Complete initiative (Iniciativa completa).
    - Any tag (Cualquiera).

As well as the inteface for CO queries, the user can specify the presentation options, but this interface does not include "Only one result per document" option, since instead of retrieving the best entry point unit, the user indicates the type of unit to retrieve.

Once the basic components of the interface have been presented, we want to establish the differences with respect to *Bricks*, defined by Zwol et al. in [100], the most similar approach found in the specialized literature. *Bricks* is a visual query formulation technique for structured document retrieval that aims at reducing the complexity of the query formulation process and required knowledge of the underlying document structure for the user. Then, we are going to consider the following natural language query to explain this approach better:

Find historical information about *revolutions* for destinations with a *constitutional monarchy* as government.

The first step is to specify the requested element of retrieval, "*Find historical information*". Next, a limited number of iterative steps are possible. The user either specifies a content-based constraint, "*about revolutions*", using the filter that is associated with the request path, or adds additional path directives to the

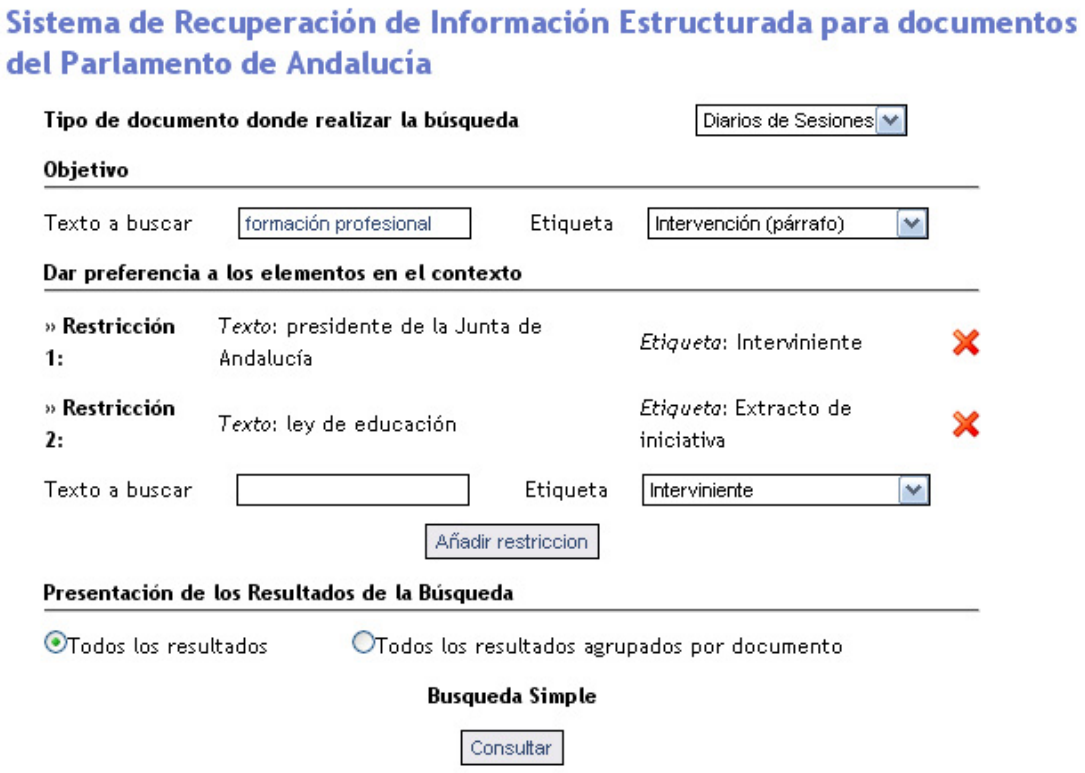


Figure 9.3: User interface for Content and Structure queries in *Seda*.

request path, "*, for destinations*". If needed the user can add one more content-based filter, and simultaneously introduce a support path to the information request. This allows the users enough flexibility to follow their intuition, and to perform intermediate checks on the specified information request.

To formulate the information need, words are added representing the relation between the bricks, such as 'in', 'with' or 'about'. These bricks consist of small building blocks where each block represents a small step in the formulation process, that needs to be completed, before another block is added to the query.

After specifying the requested element of retrieval, the user can add an about clause to the request filter, or specify additional path directives to the request path. By adding an about clause, the user can specify a text constraint to the current context and specify a sub path for this text constraint.

As a result, the composed query can be read back in natural language:

In destinations *with* government information *about* 'constitutional monarchy',
find historical information *about* revolution(s).

Then, the user interface of *Bricks* works as follows: Firstly, the user must select an element, from a list box, which will be the first in the path in the NEXI query, i.e., the root element after // ('In' in their terminology). From that element, she/he must select the retrievable element in which she/he is interested in and its associated text query ('find' and 'about' in their notation), and some restrictions, again selecting one or more pairs of element and associated query text ('with' and 'about' following their terminology). Then, the example query in NEXI language would be:

//destination[about(.//government, "constitutional
monarchy")]//history[about(., revolutions)].

We think that if a user interface of this class has to assume an almost total lack of knowledge of the structure of the documents in the collection, to leave the decision of selecting the root element of the query to the user is not a good option. In our case, this decision is totally transparent to the user, only providing the target and restrictions, which is very intuitive.

A second difference, consequence of this design, is that the construction of the NEXI query in *Bricks* is direct, as they have the first element of the NEXI query (the 'destination' element of the example query), in contrast to our approach, because with the information provided by the user, that first element has to be determined.

In the following section, the method designed to generate the NEXI query is presented.

9.2.2.1 From the visual query to the NEXI query

In order to convert a visual query to a NEXI query, the following data, extracted from the user interface, are required as input of the process: *target_element* (the desired type of elements to be retrieved), and the text query *target_text* for that target element, plus a set of pairs:

$$(context_element_1,\ context_text_1),\ldots,(context_element_N,\ context_text_N),$$

contextualizing the target element, and finally the collection *Document Type Definition* (DTD). The output is a NEXI query with the following pattern:

$$//pivot_element[context_about_list]//target_element[target_about_list].$$

Then, the translation process will have to find the different components of this NEXI query from the input data.

Once the XPath of all the different elements involved in the query are determined from the DTD, the first step is to find the *pivot_element*. This is performed extracting the common path from the set of paths composed of *target_element* and the N *context_element*'s that contain the path of *target_element*. The last element in this path is considered the *pivot_element*.

With respect to *context_about_list*, it will be composed of N about clauses joined by the 'and' operator. Inside each about clause, if the paths from the *pivot_element* and the $context_element_i$ are the same, the element restriction is '.', otherwise the element restriction is ".//" and the last element in the $context_element_i$ path. The text of the about clause will be $context_text_i$.

Finally, *target_about_list* is composed of several about clauses connected with the 'and' operator. The first about is related to the target element, containing a '.' in the element part and *target_text* in the text part. The rests of abouts come from those *context_element*'s whose paths contain the path of *target_element*. Specifically, the element part of the about clause is the last element of the path of $context_element_i$. The text part is its associated $context_text_i$.

When "any" is selected from the available labels in the list box, *target_element* equals '*'. Finally, if no context is provided, then the NEXI query is:

$$//target_element[about(., target_text)].$$

To a better understanding of the method, we are going to consider the example query of fig. 9.3. Then, we can distinguish the following elements:

- $context_element_1$: Interviniente - $context_text_1$: presidente de la Junta de Andalucía.
- $context_element_2$: Extracto de la iniciativa - $context_text_2$: ley de educación.
- $target_element$: Intervención (párrafo) - $target_text$: formación profesional.

Afterwards, we have to look for the common path of all the query elements corresponding to the *pivot_element*. As we can see in the DTD file (fig. 9.4), it corresponds to *iniciativa_desarrollo*.

Lastly, we generate the NEXI query following the previous instructions:

//iniciativa_desarrollo[about(.//intervienen_desarrollo, presidente de la Junta de Andalucía) and about(.//extracto_desarrollo, Ley de educación)]//parrafo_desarrollo[about(., formación profesional)].

In general, this is an efficient method that mainly works with string operations. The generation of the NEXI query is very fast, negligible by the user.

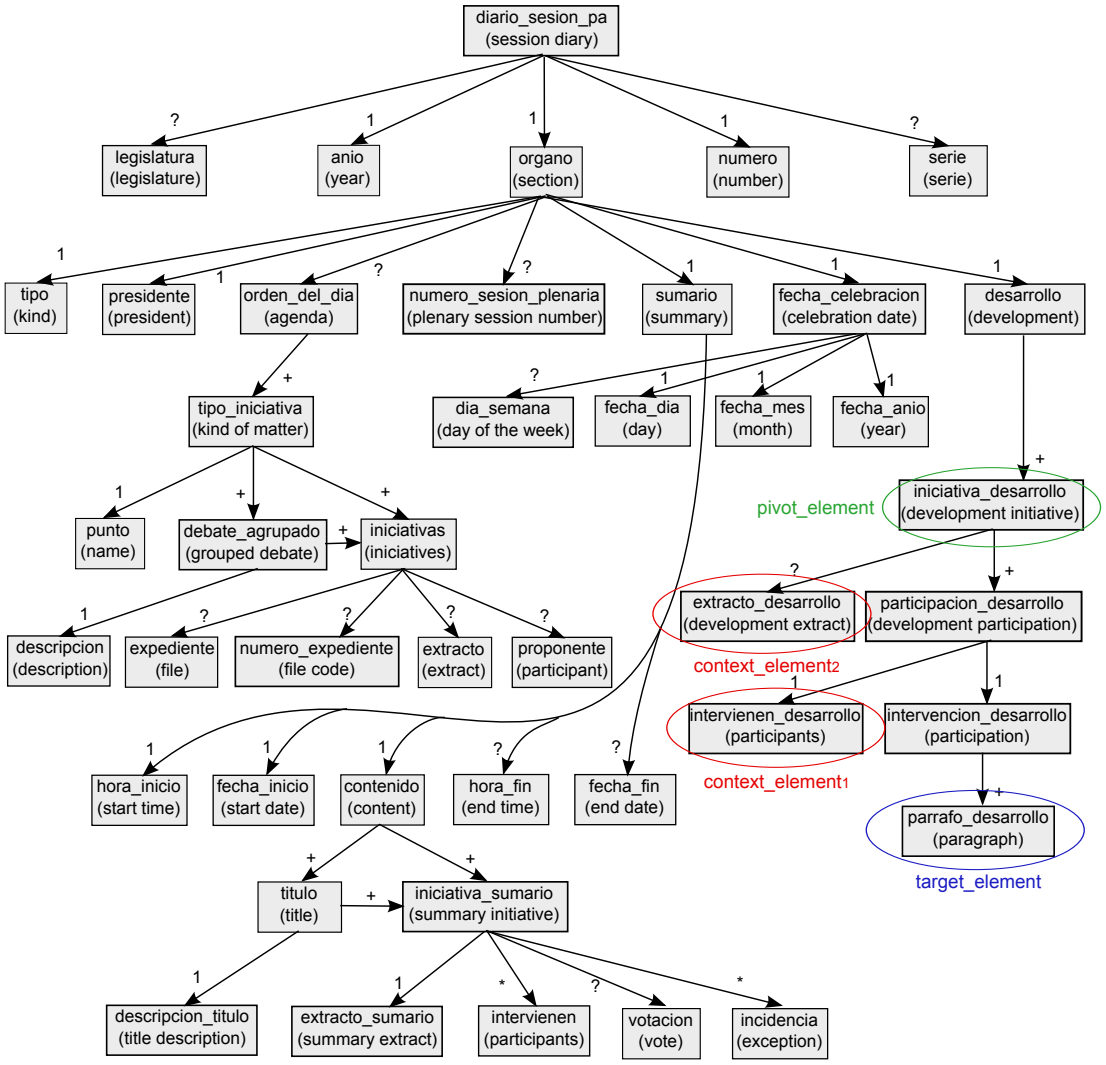


Figure 9.4: Query elements in the DTD file of the records of parliamentary proceedings.

Figure 9.5: Presentation of results in *Seda*.

9.3 Presentation of the results

Once the search engine has computed the relevance of the structural units in the collection, the results are presented on a second web page in groups of 10, sorted decreasingly in terms of relevance to the query.

For each result, a brief portion of the text of the structural unit is provided together with a link to the corresponding PDF document containing this unit and a link to the XML document visualized in HTML using XSL (eXtensible Stylesheet Language), which is a language to create style sheets to convert XML documents into many formats like HTML, highlighting the relevant units.

If there is a video for a unit, then there will also be a link to this video, so that the user will be able to watch the video segment corresponding to this structural unit. In fig. 9.5, we show an example of how the results are presented by playing a video segment when the "all the results grouped by document" option is selected.

The video player is implemented using Flash technology, as it allows videos to be played on any platform. The format of these videos is lighter than their original format, saving broadband in the video transmission. Moreover, the webpage of the video player includes two links to the previous and next videos of the session, being easier the way to access the videos of the session for the user.

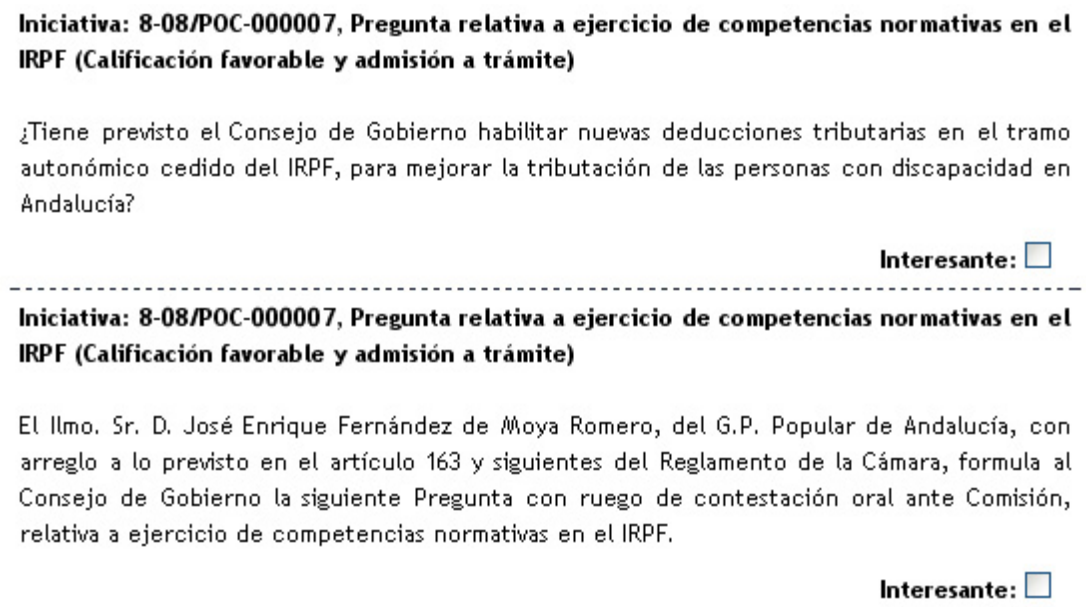


Figure 9.7: Check buttons to indicate relevant information.

After the results have been retrieved, the user can decide which of them are interesting for her/his needs. To solve this problem, a check button (*Interesante*) is shown for each result and the user only has to click on those ones satisfying the information need (see fig. 9.7). Therefore, all the Xpath of the selected results are stored in a text file. Besides, the Xpath of the non-selected results of the visited pages are stored in another file too. Both files corresponding to the Xpath of the relevant and non-relevant results are used in the RF framework.

As the results are presented in groups of 10 organized in different pages, it could be difficult for the user to remind all the selected results. So, we have created an intermediate webpage (see fig. 9.8) where the user is able to see a summary of the relevant results following the same format of the results webpage. After analyzing the summary, the user can decide to continue this selection process (clicking on "Volver página anterior"), visiting more pages or reviewing the selection she/he has done, or launch the RF framework with the relevance information (clicking on "Buscar documentos similares"), generating a new expanded query.

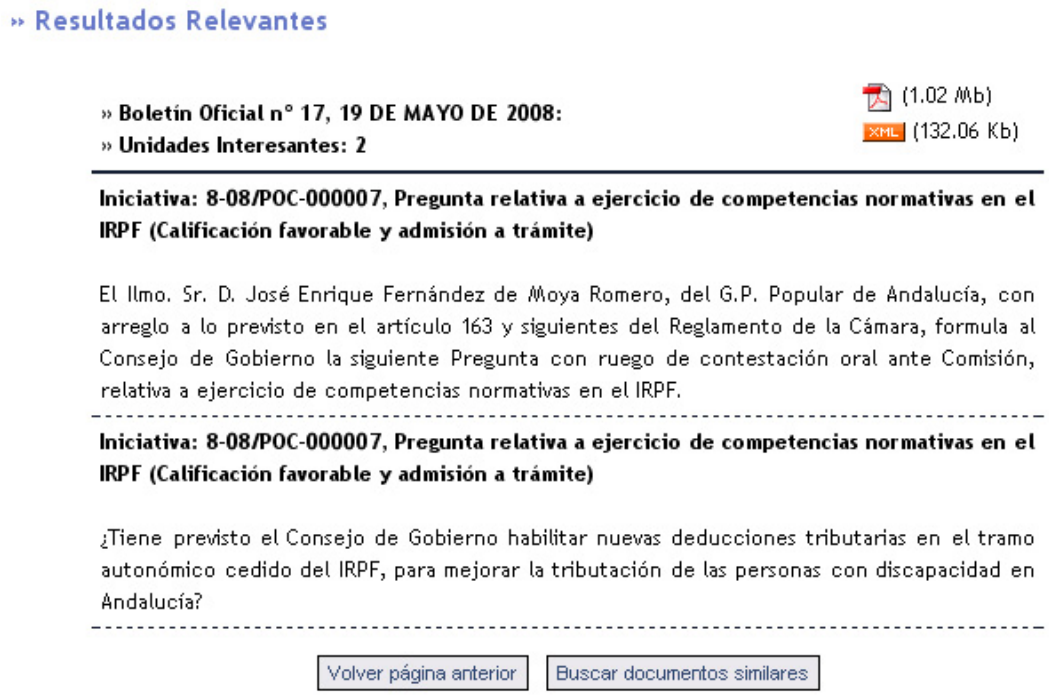


Figure 9.8: Relevant results selected by the user.

Then, this new expanded query is run into the search engine and the retrieved results are shown as we have described in the previous section.

9.5 Conclusions and future research

This chapter has presented the web application of the AP, *Seda*, to access, in an intuitive way, the collections of the AP. It is composed of several graphic user interfaces used to facilitate the formulation of CO and CAS queries by the user, without the need of knowing any XML query language and being an expert in the internal structure of the XML collection, present the results retrieved by the search engine and interact with the RF framework.

For the CO queries, the user interface is based on the use of keywords queries being possible to add several restrictions. For CAS queries, the user interface is composed of two main graphic component groups, one for specifying the target

of the CAS query and other for indicating the context or restrictions. In both cases, the user selects from a list of descriptive labels the XML elements in which she/he is interested in and input the associated text queries. With these data, a NEXI query is constructed by mean of a simple procedure, and passed to the search engine in charge of the retrieval of the relevant elements.

We think the presented interfaces are very intuitive and easy to use, facilitating the always complex process of giving expression to the user's information need.

With respect to the further research, new interfaces would be necessary in the development of personalization techniques to include the user's special characteristics and features in the search process.

Part V

Conclusions

Chapter 10

Conclusions and Future Works

This last chapter presents the general conclusions of this dissertation. We recall that the specific conclusions of the contribution and application to the Andalusian Parliament parts were previously given at the end of each corresponding chapter. Besides, a lists of publications with the results presented in this dissertation are included, together with some future works. Now, we are going to highlight the main conclusions.

In chapter 3, we show all the improvements introduced in the context-based influence diagram model implemented in Garnata. Firstly, the methodology to adapt the results retrieved by Garnata to the different INEX *adhoc* tasks from 2007 is shown: *focused*, *relevant in context* and *best in context*. Later, the chapter describes a modification of the model developed to compute the weights and utilities of the structural units of the documental collection with the objective of retrieving the most adapted structural units to the users' needs. Lastly, this chapter shows a parametric model included in the context-based influence diagram model which adjusts the degree of utility to make the system behaves more similarly to a strict or less strict AND gate for the query terms introduced in the system. The experimentation done in different years of INEX workshop, specially in 2008, for the *adhoc* tasks has certified the improvement of Garnata with these changes.

Chapter 4 presents a methodology to introduce Content and Structure queries in Garnata. This methodology could be implemented in any probabilistic structured Information Retrieval system without the need of modifying the system or

interact with it in a complex way. Experiments with the *Wikipedia* and *IEEE* collections allow us to check the efficacy of this methodology using Garnata as the structured Information Retrieval system. As a result, the right management of Content and Structure queries improves the retrieval capabilities of Garnata.

Chapter 5 shows two different methodologies for Relevance Feedback of both Content-only and Content and Structure queries. These methodologies are based on the idea of modifying the original query expanding the natural language query with new terms for the Content-only queries and expanding all the subqueries following the same criterion but keeping the structural restrictions of the original query for the Content and Structure queries. The experimentation in the different INEX workshops tells us Relevance Feedback, in general, improves the results for all the evaluation measures and the more relevance information we have, better the results. According to the number of expanded terms, the best alternative is to choose a variable number of terms depending on the query.

With respect to the part of the application of the system to the Andalusian Parliament, chapter 7 presents the XML collection used in the Andalusian Parliament and its internal organization. The methodology to transform the PDF official documents into XML format is easily adapted to the documents of any institution having a well defined structure.

Chapter 8 is focused on the presentation of two techniques to segment and synchronize the session videos with the XML documents from the Andalusian Parliament. The main idea of these two approaches is to link the XML documents of the Parliament with their corresponding videos. It permits us to retrieve both sources of information in Garnata, i.e., if the system retrieves a structural unit corresponding to a part of a speech, the user can read the text of this unit or watch the video with the intervention of the member of the parliament where the portion of the speech is reproduced.

Finally, chapter 9 shows the web application of the Andalusian Parliament, *Seda*, to be able to access the official collections (text and multimedia). In addition, it presents the way to interact with Garnata and the different tools developed in this thesis: Use of different *adhoc* tasks, CAS queries, RF, etc. The interfaces are very intuitive and easy to manage for any user, so it is unnecessary to know the way these tools work.

10.1 List of publications

The different studies included in this dissertation have been presented in the following publications:

1 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *Managing structured queries in probabilistic XML retrieval systems.* Information Processing & Management, 46(5):514–532, 2010.

2 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *Synchronising video session recordings and textual transcriptions from the Andalusian Parliament.* IADIS International Journal on Computer Science and Information Security. 4(2):120–139, 2009.

3 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín-Dancausa, A. Tagua-Jiménez, and M. C. Tur-Vigil. *An integrated system for managing the Andalusian Parliament's digital library.* Program: Electronic Library and Information Systems, 43(2):156–174, 2009.

4 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. J. Martín-Dancausa, and A. E. Romero. *An information retrieval system for Parliamentary documents.* Bayesian Networks. Chapter 12. A practical Guide to Application, pages 203–223. Ed. Wiley, 2008.

5 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. J. Martín-Dancausa. A content-based approach to relevance feedback in XML-IR for content and structure queries. Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, KDIR 2010 (Valencia - Spain), pages 418–427, 2010.

6 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. Sobre el número de términos de expansión en métodos de realimentación por relevancia para sistemas de recuperación de información estructurada. Actas del I Congreso Español de Recuperación de Información, CERI 2010 (Madrid - Spain), pages 195–206, 2010.

7 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *Content-oriented relevance feedback in XML-IR using the Garnata information retrieval system.* Lecture Notes in Artificial Intelligence, FQAS 2009 (Roskilde - Denmark), 5822:617–628, 2009.

8 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *A graphic user interface for content and structure queries in xml retrieval.* Proceedings of the third workshop on Human Computer Information Retrieval, HCIR 2009 (Washington - USA), pages 34–37, 2009.

9 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín-Dancausa, and A. E. Romero. *New utility models for the garnata information retrieval system at INEX'08.* Lecture Notes in Computer Science, INEX 2008 (Dagsthul - Germany), 5631:39–45, 2009.

10 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *Modelos de utilidad para sistemas de recuperación de información estructurada basados en modelos gráficos probabilísticos.* XIV Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2008 (Langreo - Asturias), pages 669–674, 2008.

11 L. M. de Campos, J. M. Fernández-Luna, J. M. Garcia, F. Gómez, J. H. Huete and C. J. Martín-Dancausa. *A video segmentation and annotation tool for Parliamentary recordings and transcriptions.* Multi Conference on Computer Science and Information Systems, IADIS Informatics 2008 (Amsterdam - Netherlands), pages 35-42, 2008.

12 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. *An integrated system for accessing the digital library of the Parliament of Andalusia: Segmentation, annotation and retrieval of transcriptions and videos.* Proceedings of the 8th international workshop on Pattern Recognition and Information Systems, PRIS 2008 (Barcelona - Spain), pages 38–47, 2008.

13 J. M. Fernández-Luna, J. F. Huete, M. Gómez, and C. J. Martín-Dancausa. *Development of the xml digital library from the Parliament of Andalucía*

for intelligent structured retrieval. Proceedings of the 17th international conference on Foundations of Intelligent Systems, ISMIS 2008 (Toronto - Canada), pages 417–423, 2008.

14 J. M. Fernández-Luna, J. F. Huete, M. Gómez, and C. Martín-Dancausa. *Diagramas de influencia y bibliotecas digitales: Un caso de estudio con los diarios de sesiones del Parlamento de Andalucía.* Actas del II simposio de Inteligencia Computacional, CEDI 2007 (Zaragoza - Spain), pages 345–356, 2007.

15 L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín-Dancausa, and A. E. Romero. *The Garnata Information Retrieval System at INEX'07.* Lecture Notes in Computer Science, INEX 2007 (Dagsthul - Germany), 4862:57–69, 2008.

10.2 Future works

In the following lines, we present some ideas to continue developing this work in the future.

Firstly, the Garnata filter strategy should be improved to determine in a higher level the type of unit retrieved by the structured Information Retrieval.

According to the Content and Structure queries, we have to research in the interpretation of them and take advantage of the flexibility of the noisy-OR/AND used by our structured Information Retrieval system to improve further the results.

In Relevance Feedback, it would be important to determine the number of terms used to expand the query depending on the type of query and study how to apply the negative Relevance Feedback (of the non-relevant units) for the expansion. For Content and Structure queries, it could be interesting to study if the structural restrictions of the context could be modified depending on the relevance information given by the user, although the structural restrictions of the objective would not be modified because they indicate what the user is looking up, so they should not change.

With respect to the application of the Andalusian Parliament, the future works would be based on the expansion of the official XML document collection from the first legislature, since the current collection is from the sixth legislature. On the other hand, the most part of the video synchronization of the Andalusian Parliament process is made by the user. Then, it should be improved to be more automatic. Finally, in *Seda*, we should develop the interface for Relevance Feedback in Content and Structure queries and include new interfaces for the development of personalization techniques.

Appendixes

Appendix of Chapter 4

291	//article//figure[about(.,olympian god goddess)]
292	//article//figure[about(.,renaissance painting italian flemish)]
293	//article[about(.,wifi)]//section[about(.,wifi security encryption)]
294	//article[about(.,user interface)]//section[about(.,design usability guidelines)]
295	//article[about(.,software)]//section[about(.,intellectual property) or about(.,patent license)]
304	//article[about(.,allergies)]//section[about(.,treatment)]
307	//article[about(.,islam islamic)]//section[about(.,qur'an) or about(.,prophet muhammad)]
308	//article[about(.,wedding)]//section[about(.,traditions customs)]
310	//article[about(.,novikov self-consistency principle) and about(.//section,time travel)]
313	//article[about(.,immanuel kant moral philosophy)]//section[about(.,categorical imperative)]
315	//article[about(.,spider)]//section[about(.,hunting) and about(.//p,insect)]
318	//article[about(.,atlantic ocean island)]//section[about(.,slave trade)]
327	//article[about(.,cloning animals)]//section[about(.,acceptance) and about(.,united states of america)]
331	//figure[about(.,tulips)]
334	//article[about(.,silk road)]//section[about(.,silk road china)]
337	//article[about(.,computer networks)]//section[about(.,security algorithms)]
347	//article[about(.,state machine)]//figure[about(.,mealy) or about(.,moore)]
355	//article[about(.,film)]//section[about(.,awards best actress academy award winner)]
357	//article[about(.,babylonia babylonian assyria assyriology)]//section[about(.,babylonia babylonian)]
360	//article[about(.,solar energy)]//section[about(.,domestic electricity heating)]
362	//article[about(.,nuclear power plant)]//section[about(.,effect accident)]
366	//article[about(.,fourier transform)]//p[about(.,applications)]
369	//article[about(.,pillars of hercules)]//section[about(.,mythology)]
373	//article[about(.,spy network)]//*[about(.,australia echelon)]
375	//article[about(.//p,states countries nuclear proliferation nonproliferation treaty npt) and about(.//p,weapons civilian)]
376	//article[about(.,diabetes mellitus)]//section[about(.,type 2 symptoms)]
382	//article[about(.,greek mythology) and about(.//caption,greek)]//*[about(.,aphrodite)]
386	//article[about(.,fencing)]//p[about(.,weapon)]
392	//article[about(.,australian aboriginals)]//section[about(.,stolen generation)]
399	//article[about(.,mobile phone country)]//section[about(.//table,umts) and about(.,umts)]
403	//article[about(.,analog color television)]//p[about(.,standard description)]
407	//article[about(.,football world cup)]//p[about(.,miracle of bern)]
409	//article[about(.,hybrid vehicle)]//p[about(.,fuel efficiency fuel sources model engine)]
413	//article[about(.,capital city europe)]//section[about(.,coordinates population)]

Table 1: List of the 34 INEX 2006 Content and Structure queries in NEXI format used in the experiments with the *Wikipedia* collection.

415	//article[about(.,space history)]//section[about(.,astronaut cosmonaut engineer)]
420	//article[about(.,shading models)]//section[about(.,phong shading)]
421	//article[about(.,neil gaiman novels)]//section[about(.,plot details)]
422	//article[about(.,bird) or about(.,passerine)]//p[about(.,song)]
428	//article[about(.,chinese traditional religion)]//section[about(.,taoism)]
429	//article[about(.,chinese dynasties)]
430	//article[about(.,steganography) or about(.,steganography techniques)]
431	//article[about(.,computer graphics)]//section[about(.,opengl)]
434	//article[about(.,nietzsche)]//section[about(.,book)]
444	//section[about(.//p,tcp ip)]//p[about(.,port forwarding)]
448	//article[about(.,french fifth republic)]//section[about(.,president)]
463	//article[about(.,healthy diet)]//section[about(.,diet features)]
464	//p[about(.,simpsons references)]
470	//section[about(.,operating system)]//p[about(.,page replacement policy)]
472	//article[about(.,prusik knot)]//figure[about(.,prusik)]
474	//section[about(.,engine diesel)]//p[about(.,fuel consumption)]
480	//article[about(.,computer) and about(.,monthly magazine)]
482	//article[about(.,characters of holy book ramayana) or about(.,ramayana)]
483	//article[about(.,theory) and about(.,origin universe)]
484	//article[about(.//category,artists)]//*[about(.,dutch artists paris)]
487	//article[about(.,brazil) and about(.//section,tourism)]
488	//article[about(.,football world cup fifa)]
489	//article[about(.,southern china)]//section[about(.,tourism)]
495	//article[about(.,machine learning algorithm) and about(.,machine learning theory)]
497	//article [about(.,first) and about(.,wikipedia)]
511	//article[about(.,george best)]//section[about(.,death) or about(.,died) or about(.,passed away)]
525	//*[about(.,potatoes) and about(.,paintings)]//figure[about(.,potatoes) and about(.,painting)]
526	//section[about(.,pyramids egypt) and about(.//(figure|image),pyramids)]
527	//article[about(.,walt disney land world) and about(.//(figure|image),disney land)]
530	//figure[about(.,hurricane)]
533	//(figure|image)[about(.,phone)]
537	//*[about(.,mont blanc) and about(.//(figure|image),mont blanc)]
538	//article[about(.,photographer) and about(.//(figure|image),photo)]
539	//figure[about(.//caption,self-portrait)]
542	//(figure|image)[about(.,tsunami)]
543	//(figure|image)[about(.,tux)]

Table 2: List of the 36 INEX 2007 Content nd Structure queries in NEXI format used in the experiments with the *Wikipedia* collection.

544	//article[about(.,philosophy)]//section[about(.,meaning of life)]
545	//article[about(.,dance)]//section[about(.,style)]
546	//article[about(.,history)]//section[about(.,19th century imperialism)]
550	//article[about(.,dna)]//section[about(.,test)]
555	//article[about(.//picture,amsterdam) or about(.//image,amsterdam)]
576	//article[about(.,aviation)]//p[about(.,aircraft formation)]
578	//section[about(.,childbirth tradition)]
581	//*[about(.,wine tasting) or about(.,wine)]
597	//article[about(.,database) or about(.,expert)]//section[about(.,database expert)]
598	//section[about(.,mahler symphony)]//*[about(.,song)]
600	//article[about(.,japanese culture)]//p[about(.,food)]
602	//article[about(.,dictionary)]//p[about(.,webster)]
603	//article[about(.,motors) or about(.,automobiles)]//p[about(.,tata)]
607	//article[about(.,nuclear)]//p[about(.,law) or about(.,legislation) or about(.,act)]
629	//article[about(.,film)]//section[about(.,science fiction)]
637	//article[about(.,java programming languages)]//section[about(.,applications of java)]
659	//section[about(.,technological singularity)]
668	//article[about(.,colossus code-breaking)]//section[about(.,bletchley park)]
669	//section[about(.,coin collecting)]
673	//section[about(.,intrusion detection)]

Table 3: List of the 20 INEX 2008 Content and Structure queries in NEXI format used in the experiments with the *Wikipedia* collection.

		iP[0.01]			AiP	
Query	Augm.-CAS	Base-CO	Base-CAS	Augm.-CAS	Base-CO	Base-CAS
291	0.04460	0.00871	0.04460	0.00833	0.00292	0.00833
292	0.66149	0.03105	0.66149	0.05995	0.00252	0.05995
293	1.00000	0.57261	1.00000	0.31303	0.43355	0.38455
294	1.00000	0.94037	1.00000	0.20720	0.30639	0.19421
295	0.84029	0.41597	0.74791	0.12757	0.12114	0.14363
304	0.39899	0.30393	1.00000	0.03704	0.02897	0.05400
307	0.21426	0.14873	0.13277	0.07113	0.07556	0.07129
308	0.75123	0.61193	0.64036	0.17494	0.08113	0.16735
310	1.00000	0.97749	1.00000	0.78168	0.23832	0.82698
313	0.97838	0.51579	1.00000	0.28906	0.23667	0.19598
315	0.76620	0.49489	0.71643	0.06069	0.05113	0.04409
318	0.08417	0.00336	0.03078	0.05109	0.00087	0.01841
327	1.00000	0.58446	0.31854	0.17109	0.08638	0.03195
331	0.68365	0.19140	0.68365	0.02344	0.09249	0.02344
334	0.83373	0.63629	0.99095	0.31647	0.21602	0.24332
337	0.07666	0.02895	0.11627	0.01758	0.00789	0.01943
347	0.84821	0.11736	0.68493	0.05879	0.04710	0.02962
355	0.03758	0.00762	0.03651	0.00531	0.00283	0.00555
357	0.94408	0.87974	0.97032	0.31471	0.44612	0.30204
360	0.84428	0.78447	1.00000	0.22690	0.54327	0.25231
362	0.26812	0.05146	0.06742	0.03389	0.03364	0.02744
366	0.39508	0.08766	0.61165	0.01511	0.03345	0.08299
369	0.65696	1.00000	0.65696	0.12635	0.03096	0.16079
373	0.20928	0.07189	0.07189	0.19685	0.03893	0.03893
375	0.08917	0.07989	0.09207	0.03358	0.05001	0.05317
376	0.61521	0.32973	0.61521	0.11630	0.13139	0.11324
382	0.93619	0.52754	0.52754	0.32437	0.31752	0.31752
386	0.74598	0.67702	0.85451	0.07032	0.12669	0.05494
392	0.51902	0.47099	0.60693	0.27754	0.27460	0.30169
399	1.00000	0.28935	0.76108	0.35452	0.22700	0.19310
403	0.41546	0.22920	0.32273	0.02023	0.12327	0.02144
407	1.00000	1.00000	1.00000	0.15604	0.62673	0.16864
409	1.00000	0.95235	0.63652	0.23839	0.50372	0.13596
413	0.01020	0.00425	0.00414	0.01190	0.00133	0.00030

Table 4: Detailed results per query for iP[0.01] and AiP using the 34 INEX 2006 Content and Structure queries.

	iP[0.01]			AiP		
Query	Augm.-CAS	Base-CO	Base-CAS	Augm.-CAS	Base-CO	Base-CAS
415	0.41188	0.10020	0.05003	0.03118	0.02157	0.00594
420	0.97421	0.46711	0.53599	0.27349	0.25737	0.21546
421	0.08287	0.07778	0.01844	0.00514	0.00570	0.00223
422	0.47726	0.33333	0.01886	0.15199	0.03845	0.00279
428	0.54670	0.19546	0.37765	0.19614	0.06826	0.09142
429	0.39120	0.39823	0.39120	0.17033	0.04076	0.17033
430	1.00000	0.95275	0.95271	0.63189	0.41167	0.50772
431	0.98273	0.38706	0.58617	0.34080	0.13278	0.20192
434	0.86993	0.61192	0.71744	0.06026	0.12089	0.31248
444	0.15366	0.04877	0.02276	0.01035	0.01428	0.00051
448	0.55402	0.57618	0.99219	0.06504	0.04714	0.11473
463	1.00000	0.91338	1.00000	0.56726	0.42323	0.40398
464	0.34535	0.17566	0.34535	0.02755	0.04294	0.02755
470	1.00000	0.94063	1.00000	0.22315	0.58218	0.12098
472	1.00000	0.67574	1.00000	0.01980	0.22313	0.01980
474	1.00000	0.98590	1.00000	0.16145	0.52697	0.11927
480	0.46862	0.42572	0.76274	0.10845	0.07676	0.10298
482	0.96576	0.99359	0.99359	0.37280	0.19099	0.20106
483	0.18047	0.09609	0.11841	0.11417	0.01299	0.07949
484	0.99471	0.99471	0.99471	0.15902	0.15902	0.15902
487	0.49939	0.10646	0.08869	0.04289	0.05133	0.05229
488	0.73738	0.52153	0.73738	0.16182	0.05725	0.16182
489	0.67209	0.12104	0.66923	0.10437	0.04810	0.10393
495	0.76830	0.78521	0.76830	0.13471	0.08503	0.10790
497	0.03354	0.00739	0.03354	0.02278	0.00070	0.02278
511	0.12060	0.00039	0.00055	0.04603	0.00034	0.00043
525	0.32174	0.01052	0.24511	0.03443	0.00115	0.03240
526	0.05524	0.01649	0.02773	0.00565	0.00620	0.00217
527	0.06865	0.02625	0.05442	0.03868	0.00506	0.03812
530	0.31607	0.01065	0.31607	0.01732	0.00084	0.01732
533	0.37813	0.06393	0.00000	0.00946	0.00208	0.00559
537	0.92133	0.19247	0.19247	0.38903	0.04696	0.04696
538	0.13871	0.84939	0.91789	0.01943	0.05668	0.07347
539	0.97346	0.51733	1.00000	0.28670	0.11084	0.29187
542	0.26275	0.00036	0.26275	0.05081	0.00002	0.05115
543	0.57692	0.00197	0.57692	0.20885	0.00018	0.20885

Table 5: Detailed results per query for iP[0.01] and AiP using the 36 INEX 2007 Content and Structure queries.

	iP[0.01]			AiP		
Query	Augm.-CAS	Base-CO	Base-CAS	Augm.-CAS	Base-CO	Base-CAS
544	0.11635	0.26339	0.23657	0.03966	0.02368	0.05657
545	0.84679	0.79527	0.55989	0.09644	0.14366	0.11287
546	0.58191	0.40181	0.22365	0.05055	0.07570	0.03765
550	0.85852	0.29368	0.75747	0.13366	0.05163	0.12082
555	0.03218	0.00128	0.00487	0.00687	0.00004	0.00203
576	0.00692	0.00179	0.00929	0.00240	0.00119	0.00484
578	0.09163	0.10394	0.09163	0.02428	0.01047	0.02428
581	0.95469	0.42982	0.42982	0.02158	0.02137	0.02137
597	0.96844	0.26598	0.96844	0.05662	0.00533	0.02606
598	0.75385	0.14653	0.27434	0.04669	0.11108	0.03582
600	0.79630	0.44981	0.09364	0.02791	0.07512	0.00467
602	0.78538	0.44842	0.66351	0.08588	0.07277	0.08609
603	1.00000	0.99282	1.00000	0.31977	0.21643	0.06432
607	0.99940	0.05355	0.20583	0.32018	0.02284	0.03132
629	0.37466	0.17923	0.17953	0.02943	0.01165	0.02400
637	1.00000	0.55437	0.61479	0.19169	0.21442	0.10788
659	0.79146	0.89460	0.79146	0.15102	0.10743	0.15102
668	0.84929	0.99899	0.99439	0.21688	0.25280	0.26065
669	0.95996	0.75392	0.95996	0.13869	0.02914	0.13869
673	0.99838	0.97130	0.99838	0.35747	0.27335	0.35747

Table 6: Detailed results per query for iP[0.01] and AiP using the 20 INEX 2008 Content and Structure queries.

Times	iP[0.01]		AiP	
Augm.-CAS	Base-CO	Base-CAS	Base-CO	Base-CAS
better	78	44	60	50
worse	10	18	29	25
equal	2	28	1	15

Table 7: Number of times that the Augmented CAS is better/worse/equal that Base-CO and Base-CAS methods for iP[0.01] and AiP, using the 90 INEX-Wikipedia Content and Structure queries.

62	//article[about(.,security biometrics) and about(.//sec,facial recognition)]
63	//article[about(.,digital library) and about(.//p,authorization access control security)]
64	//article[about(.,hollerith)]//sec[about(.,DEHOMAG)]
67	//article//fm[about(.//(tig|abs),software architecture)]
68	//article[about(.,Smalltalk) or about(.,Lisp) or about(.,Erlang) or about(.,Java)]// bdy//sec[about(.,garbage collection algorithm)]
69	//article//bdy//sec[about(.//st,information retrieval)]
70	//article[about(.//fm//abs,information retrieval digital libraries)]
71	//article[about(.,formal methods verify correctness aviation systems)]// bdy//*[about(.,case study application model checking theorem proving)]
72	//article[about(.//fm//au//aff,United States of America)]//bdy//*[about(.,weather forecasting systems)]
74	//article[about(.,video streaming applications)]//sec[about(.,media stream synchronization) or about(.,stream delivery protocol)]
75	//article[about(.,Petri net) and about(.//sec,formal definition) and about(.//sec, algorithm efficiency computation approximation)]
77	//article[about(.//sec,reverse engineering)]//sec[about(.,legal) or about(.,legislation)]
78	//vt[about(.,Information Retrieval student)]
79	//article[about(.,XML) and about(.,database)]
80	//article//bdy//sec[about(.,clock synchronization distributed systems)]
81	//article[about(.//p,multi concurrency control) and about(.//p,algorithm) and about(.//fm//atl,databases)]
82	//article[about(.,handwriting recognition) and about(.//fm//au,kim)]
83	//article//fm//abs[about(.,data mining frequent itemset)]
84	//p[about(.,overview distributed query processing join)]
86	//sec[about(.,mobile electronic payment system)]
89	//article[about(.//bdy,clustering vector quantization fuzzy k-means c-means)]//bm//bb [about(.,vector quantization fuzzy clustering k-means c-means) and about(.//pdt,1999)]
90	//article[about(.//sec,trust authentication electronic commerce e-commerce e-business marketplace)]//abs[about(.,trust authentication)]

Table 8: List of the 22 INEX 2003 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection.

127	//sec//(p|fgc)[about(.,Godel Lukasiewicz other fuzzy implication definitions)]
128	//article[about(.,intelligent transport systems)]//sec[about(.,on-board route planning navigation system for automobiles)]
129	//article[about(.//atl,new book review bookshelf)]//sec[about(.,database data warehouse)]
130	//article[about(.//p,object database)]//p[about(.,version management)]
131	//article[about(.//au,Jiawei Han)]//abs[about(.,data mining)]
132	//article[about(.//abs,classification)]//sec[about(.,experiment compare)]
135	//article[about(.//atl,summaries)]//sec[about(.,Internet security) or about(.,network security)]
136	//bib[about(.,text categorisation) and about(.,Support Vector Machines SVM)]
137	//article[about(.//abs,digital library) or about(.//ip1,digital library)]
141	//article[about(.,java)]//sec[about(.,implementing threads)]
142	//abs[about(.,database access using perl)]
145	//article[about(.,information retrieval)]//p[about(.,relevance feedback)]
149	//article[about(.,animation)]//bdy//sec[about(.//st,inverse kinematics)]
150	//article[about(.//(abs|kwd),genetic algorithm)]//bdy//sec[about(.,simulated annealing)]
151	//article[about(.,web search engine)]//sec[about(.,vector space model)]
152	//article//p[about(.,linux word processor office programs)]
153	//article//bm//vt[about(.,phD student) OR about(.,phD final)]
155	//article[about(.//p,self organising feature map) and about(.//fm//yr,2000)]//fig[about(.//fgc,self organising map)]
156	//article[about(.//abs,spatial join)]//bdy//sec[about(.,performance evaluation)]

Table 9: List of the 19 INEX 2004 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection.

202	//article[about(.,ontologies)]//sec[about(.,ontologies case study)]
203	//sec[about(.,code signing verification)]
208	//article[about(.,Artificial Intelligence history)]
210	//article//(abs|sec)[about(.,multimedia document models content authoring)]
216	//sec[about(.,multimedia retrieval system architecture) or about(.//fig,multimedia retrieval architecture)]
219	//sec[about(.,learning object granularity)]
222	//article[about(.,bussiness strategies)]//sec[about(.,eletronic commerce e-commerce)]
223	//article[about(.//sec,wireless ATM multimedia)]
229	//article[about(.//bdy,latent semantic analysis latent semantic indexing)]
230	//article//sec[about(.,brain research differential geometry)]
232	//article[about(.//abs,Dempster-Shafer theory)]//sec[about(.,Dempster Shafer database experiment)]
233	//article[about(.//bdy,synthesizers) and about(.//bdy, music)]
234	//article[about(.//atl,upcoming events) or about(.//atl,call for papers)]//sec[about(.,multimedia conference workshop)]
236	//article[about(.,machine translation approaches)]

Table 10: List of the 14 INEX 2005 Content and Structure queries in NEXI format used in the experiments with the *IEEE* Computer Society collection.

Bibliography

[1] W3c. xml path language (xpath) version 2.0., 2007. 32, 96

[2] A. V. Aho, J. D. Ullman, and J. E. Hopcroft. *Data Structures and Algorithms*. Addison Wesley, 1983. 21

[3] M. S. Ali, M. Consens, X. Gu, Y. Kanza, F. Rizzolo, and R. Stasiu. Efficient, effective and flexible xml retrieval using summaries. *Lecture Notes in Computer Science*, 4518:89–103, 2007. 109

[4] N. Apostoloff and A. W. Fitzgibbon. Automatic video segmentation using spatiotemporal t-junctions. In *Proc. BMVC*. Citeseer, 2006. 202

[5] P. Arvola, J. Kekäläinen, and M. Junkkari. Query evaluation with structural indices. *Lecture Notes in Computer Science*, 3977:134–145, 2006. 109

[6] Y. Avrithis. A stochastic framework for optimal key frame extraction from mpeg video databases. *Computer Vision and Image Understanding*, 75(1–2):3–24, 1999. 207

[7] M. Azevedo, L. Pantuza, and N. Ziviani. A universal model for xml information retrieval. *Lecture Notes in Computer Science*, 3493:311–321, 2005. 108

[8] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern Information Retrieval*, volume 463. ACM Press / Addison-Wesley, 1999. 19, 22, 23, 28, 30, 67, 169

[9] A. Bookstein. Relevance. *Journal of the American Society for Information Science*, 30(5):269–73, 1979. 22

[10] J. Calic and E. Izquierdo. Efficient key-frame extraction and video analysis. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, pages 28–33. IEEE Computer Society, 2002. 207

[11] J. Calic, S. Sav, E. Izquierdo, S. Marlow, N. Murphy, and A. E. OĆonnor. Temporal video segmentation for real-time key frame extraction. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 4, pages IV–IV. IEEE Computer Society, 2002. 202

[12] J. Calic and B. Thomas. Spatial analysis in key-frame extraction using video segmentation. In *Proceedings of the Workshop on Image Analysis for Multimedia Interactive Services*. Citeseer, 2004. 207

[13] F. Camastra and A. Vinciarelli. Video segmentation and keyframe extraction. *Advanced Information and Knowledge Processing*, 3:413–430, 2007. 195, 202

[14] N. Chang. Data manipulation in an xml-based digital image library. *Program: Electronic Library and Information Systems*, 39(1):62–72, 2005. 162

[15] Y. K. Chang, C. Cirillo, and J. Razon. Evaluation of feedback retrieval using modified freezing, residual collection & test and control groups. *The SMART Retrieval System: Experiments in Automatic Document Processing, chapter 17*, pages 355–370, 1971. 136

[16] Y. Chiaramella. Information retrieval and structured documents. *Lectures on information retrieval*, pages 286–309, 2001. 5

[17] G. F. Cooper. Probabilistic inference using belief networks is np-hard. *Artificial Intelligence*, 42(1):393–405, 1990. 52

[18] F. Crestani, L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. A multi-layered bayesian network model for structured document retrieval. *Lecture notes in artificial intelligence*, 2711:74–86, 2003. 50

[19] C. J. Crouch, A. Mahajan, and A. Bellamkonda. Flexible retrieval based on the vector space model. *Lecture Notes in Computer Science*, 3493:292–302, 2005. 106

[20] C. J. Crouch, A. Mahajan, and A. Bellamkonda. Flexible xml retrieval based on the extended vector model. *Lecture Notes in Computer Science*, 3493:292–302, 2005. 123

[21] L. M. de Campos, J. M. Fernández-Luna, J. Huete, and S. Linares. Alhaken: An information retrieval system for the session diaries of the parliament of andalucía. In *Proceedings of the IADIS e-Society conference*, pages 133–137, 2006. 187, 188

[22] L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. The bnr model: foundations and performance of a bayesian network-based retrieval model. *International Journal of Approximate Reasoning*, 34(2-3):265–285, 2003. 7, 57

[23] L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Using context information in structured document retrieval: an approach based on influence diagrams. *IP&M*, 40(5):829–847, 2004. 49, 50, 58, 59, 60, 66

[24] L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Improving the context-based influence diagram model for structured document retrieval: removing topological restrictions and adding new evaluation methods. *Advances in Information Retrieval*, pages 215–229, 2005. 65

[25] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. Martín-Dancausa. Content-oriented relevance feedback in xml-ir using the garnata information retrieval system. *Lecture notes in artificial intelligence*, 5822:617–628, 2009. 122

[26] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín-Dancausa, and A. E. Romero. The garnata information retrieval system at inex'07. *Lecture Notes in Computer Science*, 4862:57–69, 2008. 85, 110

[27] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. Martín-Dancausa, A. Tagua-Jiménez, and M. C. Tur-Vigil. An integrated system for managing the andalusian parliamentś digital library. *Program: electronic library and information systems*, 43(2):156–174, 2009. 224

[28] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and C. J. Martín-Dancausa. A content-based approach to relevance feedback in xml-ir for content and structure queries. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, pages 418–427, 2010. 122

[29] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, C. J. Martín-Dancausa, and A. E. Romero. *An information retrieval system for parliamentary documents*, chapter 12, pages 203–223. Wiley, Chichester, 2008. 85

[30] H. Déjean and J. L. Meunier. A system for converting pdf documents into structured xml format. *Lecture Notes in Computer Science*, 3872:129–140, 2006. 186, 187, 188

[31] J. M. Fernández-Luna. *Modelos de Recuperación de Información Basados en Redes de Creencia*. PhD thesis, Universidad de Granada, 2001. 19

[32] I. K. Fourati, M. Tmar, and A. B. Hamadou. Structural relevance feedback in xml retrieval. *Lecture Notes in Artificial Intelligence*, 5822:168–178, 2009. 123

[33] M. Fowler, K. Scott, et al. Uml distilled. 1998. 19

[34] S. French. Decision theory. an introduction to the mathematics of rationality. 1986. 50

[35] N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas. Inex: Initiative for the evaluation of xml retrieval. In *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval*, 2002. 95

[36] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai. Advances in xml information retrieval and evaluation. In *Proceedings of the 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005*. Springer, 2006. 95

[37] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik. Advances in xml information retrieval. In *Proceedings of the 3rd International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004*. Springer, 2005. 95

[38] N. Fuhr, M. Lalmas, and A. Trotman. Comparative evaluation of xml information retrieval systems. In *Proceedings of the 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006*. Springer, 2007. 95

[39] N. Fuhr, M. Lalmas, and A. Trotman. Focused access to xml documents. In *Proceedings of the 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007*. Springer, 2008. 95

[40] N. Fuhr, S. Malik, and M. Lalmas. In *Proceedings of the 2nd International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2003*. Springer, 2003. 95

[41] M. Gastaud and M. Barlaud. Video segmentation using active contours on a group of pictures. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, pages II–81. IEEE Computer Society, 2002. 202

[42] S. Geva. Gpx – gardens point xml information retrieval at inex 2004. *Lecture Notes in Computer Science*, 3493:211–223, 2005. 106

[43] S. Geva, J. Kamps, and A. Trotman. Inex 2008 workshop pre-proceedings. 92

[44] A. Godil. Visa: Video segmentation and annotation. In *Proceedings of the UPA Conference*, pages 81–84. Citeseer, 2004. 202

[45] A. Graves and M. Lalmas. Video retrieval using an mpeg-7 based inference network. In *Proceedings of the 25th ACM SIGIR conference*, pages 339–346. New York: ACM, 2002. 50

[46] A. Gurcan, Y. Khramov, A. Kroogman, and P. Mansfield. Converting pdf to xml with publication-specific profiles. In *Proceedings of the XML Conference and Exposition*, 2003. 186, 187, 188

[47] A. Hanjalic, R. Lagendijk, and J. Biemond. Automated segmentation of movies into logical story units. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(4):580–588, 1999. 202

[48] M. R. B. Hardy and F. D. Brailsford. Mapping and displaying structural transformations between xml and pdf. In *Proceedings of the 2002 ACM symposium on Document engineering*, pages 95–102. ACM Press, 2002. 186, 188

[49] D. Hiemstra, H. Rode, R. van Os, and J. Flokstra. Pf/tijah: text search in an xml database system. In *Proceedings of the 2nd International Workshop on Open Source Information Retrieval*, pages 12–17. ACM, 2006. 115

[50] L. Hlaoua, K. Sauvagnat, and M. Boughanem. Structure-oriented relevance feedback in xml retrieval. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 780–781. ACM, 2006. 124

[51] D. A. Hull. Stemming algorithms: A case study for detailed evaluation. *J. Am. Soc. Inform. Sci.*, 47(1):70–84, 1996. 20

[52] A. Jain and S. Chaudhuri. A fast method for textual annotation of compressed video. In *Proceedings of the Third Indian Conference on Computer Vision, Graphics & Image Processing*. Citeseer, 2002. 202

[53] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001. 50, 53

[54] J. Kamps, M. Marx, M. de Rijke, and B. Sigurbjörnsson. Articulating information needs in xml query languages. *ACM T. Inform. Syst.*, 24(4):407–436, 2006. 117, 118

[55] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. Inex 2007 evaluation measures. *Lecture Notes in Computer Science*, 4862:24–33, 2008. 43, 110, 114

[56] G. Kazai and M. Lalmas. Inex 2005 evaluation metrics. *Lect. Notes Comput. Sc.*, 3977:16–29, 2006. 117

[57] G. Kazai, M. Lalmas, N. Fuhr, and N. Gövert. A report on the first year of the initiative for the evaluation of xml retrieval (inex'02). *J. Am. Soc. Inf. Sci. Technol.*, (6):551–556, 2004. 116

[58] H. Kim and C. Choi. Xml: how it will be applied to digital library systems. *The Electronic Library*, 18(3):183–9, 2000. 162

[59] V. Kobla, D. Doermann, K. I. Lin, and C. Faloutsos. Compressed domain video indexing techniques using dct and motion vector information in mpeg video. In *Proc. of the SPIE Conference on Storage and Retrieval for Still Image and Video Databases V*, volume 3022, pages 200–211. Citeseer, 1997. 202

[60] M. Lalmas. Xml retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–111, 2009. 33, 37

[61] R. Larson. Cheshire ii at inex'04: Fusion and feedback for the adhoc and heterogeneous tracks. *Lecture Notes in Computer Science*, 3493:322–336, 2005. 106

[62] L. Liu and G. Fan. Combined key-frame extraction and object-based video segmentation. *IEEE Transactions On Circuits And Systems For Video Technology*, 15(7):869–884, 2005. 202

[63] Y. Lu, W. Gao, and F. Wu. Automatic video segmentation using a novel background model. In *Proceedings of the 2002 IEEE International Symposium on Circuits and Systems*, pages 807–810. IEEE Computer Society, 2002. 202

[64] S. Malik, M. Lalmas, and N. Fuhr. Overview of inex 2004. *Lect. Notes Comput. Sc.*, 3493:1–15, 2005. 117

[65] C. D. Manning, P. Raghavan, H. Schütze, and E. Corporation. *Introduction to Information Retrieval*, volume 1. Cambridge University Press, UK, 2008. 16, 25, 26, 31

[66] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computer Machinery*, 7(3):216–244, 1960. 22

[67] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for xml retrieval. *Lecture Notes in Computer Science*, 3493:73–84, 2005. 108

[68] Y. Mass and M. Mandelbrod. Relevance feedback for xml retrieval. *Advances in XML Information Retrieval*, pages 303–310, 2005. 123

[69] V. Mihajlovic, G. Ramirez, A. P. de Vries, D. Hiemstra, and H. E. Blok. Tijah at inex 2004 modeling phrases and relevance feedback. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004*, pages 276–291. Springer, 2004. 123

[70] S. Mohan and A. Sengupta. Docbase – the inex evaluation experience. *Lecture Notes in Computer Science*, 3493:261–275, 2005. 108

[71] S. H. Myaeng, D. H. Jang, M. S. Kim, and Z. C. Zhoo. A flexible model for retrieval of sgml documents. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145. New York: ACM, 1998. 50

[72] H. Pan, A. Theobald, and R. Schenkel. Query refinement by relevance feedback in an xml retrieval system. *Conceptual Modeling–ER 2004*, pages 854–855, 2004. 123

[73] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. 50, 103, 104

[74] J. Pehcevski, J. A. Thom, S. M. M. Tahaghoghi, and A. Vercoustre. Hybrid xml retrieval revisited. *Lecture Notes in Computer Science*, 3493:153–167, 2005. 109

[75] D. Petrelli and D. Auld. An examination of automatic video retrieval technology on access to the contents of an historical video archive. *Program: Electronic Library and Information Systems*, 42(2):115–36, 2008. 163

[76] B. Piwowarski, G. Faure, and P. Gallinari. Bayesian networks and inex. In *INEX 2002 Workshop Proceedings*, pages 7–12. Citeseer, 2002. 50

[77] S. E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *J. Am. Soc. Inform. Sci.*, 27(3):129–146, 1976. 24

[78] J. J. Rocchio. *Relevance feedback in information retrieval*. Prentice-Hall, 1971. 19, 122

[79] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145, 2003. 122

[80] E. Sáez, J. M. Gonzalez, J. M. Palomares, J. I. Benavides, and N. Guil. New edge-based feature extraction algorithm for video segmentation. In *IS&T/SPIE Symposium Proceedings, Image and Video Communications and Processing*, volume 5022, pages 861–872. Citeseer, 2003. 202

[81] G. Salton and M. Lesk. The smart automatic document retrieval systemsan illustration. 8(6):391–398, 1965. 28

[82] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 23

[83] T. Saracevic. Relevance: A review and a framework for thinking on the notion in information science. *Journal of the American Society for Information Science (JASIS)*, 26(6):321–343, 1975. 22

[84] K. Sauvagnat and M. Boughanem. Using a relevance propagation method for adhoc and heterogeneous tracks at inex 2004. *Lecture Notes in Computer Science*, 3493:337–348, 2005. 106

[85] R. Schenkel and M. Theobald. Structural feedback for keyword-based xml retrieval. *Lecture Notes in Computer Science*, 3936:326–337, 2006. 123

[86] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. Amsterdam: North-Holland, 1983. 104

[87] M. Seadle and E. Greifendeder. Defining a digital library. *Library Hi Tech*, 25(2):169–73, 2007. 157

[88] R. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986. 50, 54

[89] R. D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604, 1988. 53

[90] T. Shimizu, N. Terada, and M. Yoshikawa. Development of an xml information retrieval system for queries on contents and structures. In *Proceedings of the Second International Conference on Informatics Research for Development of Knowledge Society Infrastructure*, pages 161–168. IEEE Computer Society, 2007. 108

[91] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Processing content-oriented xpath queries. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 371–380. ACM Press, 2004. 107, 114, 117, 118

[92] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. The university of amsterdam at inex 2004. In N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors, *Proceedings of the 3rd International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004*, pages 104–109. Springer, 2004. 123

[93] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. Mixture models, overlap, and structural hints in xml element retrieval. *Lecture Notes in Computer Science*, 3493:196–210, 2005. 107

[94] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001. 5

[95] M. Theobald, A. Broschart, R. Schenkel, S. Solomon, and G. Weikum. Topx – adhoc track and feedback task. *Lecture Notes in Computer Science*, 4518:233–242, 2007. 108

[96] A. Trotman and M. Lalmas. Why structural hints in queries do not help xml-retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 711–712. ACM Press, 2006. 118

[97] A. Trotman and B. Sigurbjörnsson. Narrowed extended xpath i (nexi). *Lecture Notes in Computer Science*, 3493:16–40, 2005. 36, 96, 119, 220

[98] C. J. van Rijsbergen. *Information retrieval*, volume 2. 1979. 24, 46

[99] R. van Zwol. b^3-sdr and effective use of structural hints. *Lecture Notes in Computer Science*, 3977:146–160, 2006. 106, 107, 117

[100] R. van Zwol, J. Baas, H. van Oostendorp, and F. Wiering. Bricks: The building blocks to tackle query formulation in structured document retrieval. 3936:314–325, 2006. 226

[101] J. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structured queries in bayesian networks. *Lecture Notes in Computer Science*, 3493:100–112, 2005. 106

[102] F. Weigel, K. Schulz, and H.Meuss. Ranked retrieval of structured documents with the s-term vector space model. *Lecture Notes in Computer Science*, 3493:238–252, 2005. 108

[103] T. Westerveld, H. Rode, R. van Os, D. Hiemstra, G. Ramirez, and V. M. A. de Vries. Evaluating structured information retrieval and multimedia retrieval using pf/tijah. *Lecture Notes in Computer Science*, 4518:104–114, 2007. 115

[104] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes*. Morgan Kaufmann, San Francisco, 1999. 21, 67

[105] A. Woodley, S. Geva, and S. L. Edwards. Comparing xml-ir query formation interfaces. *Australian Journal of Intelligent Information Processing Systems*, 9(2):64–71, 2006. 118

[106] R. Yeates. An xml infrastructure for archives, libraries and museums: resource discovery in the covax project. *Program: Electronic Library and Information Systems*, 36(2):72–88, 2002. 162

[107] B. Yildiz. *Information Extraction - Utilizing Table Patterns*. PhD thesis, 2004. 188

[108] N. Zhang. Probabilistic inference in influence diagrams. *Computational Intelligence*, 14(4):475–497, 1998. 54