

# The structure of documents in the scope of information retrieval: proposals for its compression, indexing and retrieval\*

Joaquín Adiego Rodríguez  
jadiego@infor.uva.es

July 29, 2005

## 1 Introduction and Aims

The classical IRS are used to retrieve documents based on their contents. These systems have features which help the user to manipulate large text collections. These kind of systems usually deal with text and occasionally images, nevertheless the greater part of the features offered by a traditional information retrieval system are directed to indexing, processing and retrieval of atomic entities. Traditionally these entities have been associated with text documents.

Nowadays the number of information retrieval systems which deal with collections of semistructured data have been increased exponentially. The reason is the massive use of Standard Markup Languages (SGML [ISO86] and XML [BPSM00, Bra00]<sup>1</sup>) to represent textual information.

As a rule, the features previously mentioned are oriented to the treatment of text without considering other aspects related to structural information. It can be seen when the user is looking for a set of terms, the system returns an ordered list of documents without taking into account structural aspect neither when the query is made nor in returning the results. Sometimes the user needs features for searching and retrieving data which bear in mind the document structure. For example, in legal information retrieval systems it can be very useful for the user to be able to include structural restrictions in the query [HMQS96]. In legal environment, the user usually makes use of the structure to find the wanted documents. It is due to the fact that the legal information is normally organised in a structural way (titles, articles, etc). It is more, the domain in this kind of information is composed by

---

\*English abstract of “La estructura de los documentos en el ámbito de recuperación de información: propuestas para su comprensión, indexación y recuperación”.

<sup>1</sup>XML 1.1 at <http://www.w3.org/TR/xml11>

different types of documents, such as laws, rules, codes, etc. . . each one has a different structure. The use of XML in numerous contexts has caused a considerable interest in exploiting the document structure in the search and retrieval processing. It can be applicated to digital libraries [BVNF98].

The design and implementation of IRS can be affected in different ways by including the structural information of the documents. First of all, the indexing process has to consider the structure to allow the user to search using content and structure. Secondly the retrieval process has to use the content and the structure to calculate the document relevance. Finally the interface has to allow the user to use the structure in the searching process [VFC02] since the query by content and structure can only be performed if the user is able to specified in the query what is searching and where is searching it. In this case, the *what* means the content and the *where* refer to the document structure. In this case, the retrieved results are parts of the document delimited by tags of the same type.

As general rule, the IRS operate on huge quantities of textual information, this means two important things, first of all, large memory is needed to allocate the information and secondly the retrieval time in the retrieval process may not be acceptable. To solve these problems data compression techniques and indexing strategies can be used respectively. Data compression reduces the amount of space needed in disk by performing an appropriated encoding. The indexing strategies are used to supply quick access to stored documents, it is similar to the alphabetical index in a traditional book, which provides a quick access to the pages where the mentioned concepts appear.

Besides, the compression of large document collections not only reduces the amount of space needed to store the data in disk, but also it reduces the time used in the retrieval in the IRS in most cases. The improvement in process time are achieved thanks to the reduction in the disk access time employed in accessing the compressed text. Over the last decades the speed of the processor has been increased a lot more than the disk transfer speed so it is not crazy to think about in changing disk access time for decompression time or in other words, disk access time for process time. On the other hand, since same years ago, the recent researches over direct search over compressed text, (for example, to search in the compressed text without previous decompressing) let us to consider as very accurate the option of compressing the text, which means the use of less space, and to search in that compressed text, achieving that the searches are performed more quickly than in the text without compression [WMB99, ZMNBY00].

## 2 Compression and Information Retrieval

The fact that these compressed texts are going to be used in IRS, brings up some requirements which make us to reject some compression meth-

ods. One of the most important requirements is the need to access the text in a random way without previously decompression. This requirement makes us to rule out the greater part of the adaptive methods like those based in Ziv-Lempel [ZL77, ZL78] and the arithmetic coding. On the other hand, semiadaptive models like Huffman [Huf52] provided little compression. Nevertheless, when texts written in natural language are going to be compressed, it has been proved to be an excellent election considering as source symbols, words instead of characters [Mof89]. The reason is that words show the true entropy of the text much better than characters do [BCW90]. The use of a model of words together with a Huffman's coder provides us with compression rates near to 25%, due to the biased distribution of words. These rates are considerable much better than those obtained with classical adaptive models. These results get slightly worse if we use a byte oriented Huffman codification (in this codification each symbol in the source text is encoded as a sequence of bytes instead of bits). Although using this method the compression and search speed is higher, which is essential and desirable in IRS of compressed text, the compression rate is up 30%. Finally, if the alphabet and the vocabulary of the collection of texts coincide, the search will be efficient and highly sophisticated, this will be applied to both sequential search and compressed inverted indexes of the texts [WMB99, NMN<sup>+</sup>00, MNZB00, ZMNBY00, MW01].

A text written in natural language is not only composed of words but also characters of punctuation, delimiters and special characters. A delimiter is a sequence of characters between two consecutive words. Bentley et al. [BSTW86] proposes to compress texts using two disjoint alphabets of symbols: one for words and another for delimiters. The encoders using this model must consider texts as a strict sequence of two independent sources of data and code both separately. Once we have decided that the text starts with a word or a delimiter it is known that after a word it comes a delimiter and vice versa. This model is known as *separated alphabets model*.

The separated alphabets model does not take into account the fact that after a word in most cases it comes a delimiter composed of a single blank space. Since about 70% of the delimiters which appear in a text are a single blank space [Mof89], Moura et al. [MNZB00] proposes a new data model which uses a single alphabet to code both words and delimiters, and it represents the delimiter made up by a single blank space in a implicit way. This model is known as the *spaceless model* and it supposes that after each decoded word a blank space will appear, unless the next symbol to be decoded will be a delimiter.

Classical techniques and methods do not take into account the structure of the documents and if searches are allowed, they will search exclusively in the content. Nowadays, some proposals try to take advantage of the structure of the documents to improve rate compression. *XMill* is a compression method which takes into account the structure of the documents when com-

pressing and decompressing. It was developed in AT&T laboratories [LS00]. XMill is a compressor of XML data and it uses *zlib* as main compression engine. Its main advantages are that it keeps the same levels of compression and speed of *gzip* and it does not need an information scheme neither to compress nor to decompress. Since it is a tool exclusively designed to store and exchange XML documents, it does not broach the search in structure elements although it uses the structure in the compression process. Another compressor specifically design for XML is *XGrind* [TH02] which can perform queries about compressed documents but it takes no advantage of the structure.

There are other solutions to compress XML data based on the use of PPM encoders, which take advantage of the structure. An example is *XMLPPM* [Che01] which is an adaptive compressor which uses different PPM models. XMLPPM uses a ESAX parser (a variant of SAX parser) to obtain the different parts (tag's names, the name and value of the tag attributes, text, etc. . . ) of the document. Each part is encoded using a different PPM Model. XMLPPM is an adaptive compressor and performs neither random searches nor random accesses to the compressed text.

IRS needs to find in an efficient way, a particular term in the texts of the collection, so it is usual to index the documents of the collection, and the most commonly and simple technique is the inverted index [BYRN99, WMB99]. As a general rule, an inverted index consist of a vector which contains every different term of the collection (vocabulary) in lexicographical order and for each term of the index a list, which contains all the positions of the text or documents where the term appears, is stored (occurrence list). The granularity of an index is defined by the entities referenced in the occurrence list. Using the granularity, an index can be classified as inverted index addressing words, documents or blocks, if the list of occurrences makes references to a position of a text, document or a block of fixed size, where the word appears. Irrespective of the kind of granularity of the index, there are different compression methods [MNZ97, WMB99, MZ00]. It is usual to use a coding by differences together with variable-length representation of integer techniques [Eli75].

At the beginning, indexing techniques were thought to index text without structure and compression. The adaptation of classical techniques to indexing compressed text using a static or semiadaptive model are relatively simple, but on the other hand there is no a clear indexing technique which allows to index semistructural documents by content and structure at the same time. Below some proposals are discussed.

Young Kye Lee et al. Propose different indexes based on B+ trees which allow to find words inside elements of structure [LYYB96]. A tree represents the hierarchical structure of a document and each node of the tree represents a particular element of the structure. Each node of the tree is represented in the index as a unique integer, so it will be necessary to have some kind

of supplementary mechanism which allows associating the identifier with the type of structure element which it represents.

On the other hand, Navarro and Baeza-Yates propose [NBY97] a generic model in which *proximal nodes* are used to perform queries by structure and content, and avoid the previously aforementioned problem restricting the expressiveness of the query language.

There are numerous IRS which successfully incorporate inverted indexes into compress documents. For example *MG*, Managing Gigabytes [WMB99], is a versatile, general purpose open source software which compresses and indexes text, scanned documents and images. MG uses a Huffman-like code, called Huffword [ZM95, MT97] to compress text which makes it to achieve a competitive level of compression. It also keeps the index compress obtaining acceptable time response. MG can compress semistructural documents processing the tag as a word, but it does not take into account the structure of the document in the retrieval process.

### 3 Aims of the Thesis

Classical retrieval systems process documents as atomic entities, they are compressed, indexed and retrieved as indivisible. But it seems reasonable that modern IRS must be able to handle more complex documents such as documents written in SGML or SML format. New representation standards for formatted documents force us to design and implement new models and tools to work with the structure of the documents. It means that documents will no longer be considered atomic entities but as a set of aggregated and interrelated objects which need to be compressed, indexed, recovered and displayed integrated or separately in accordance with the user's need.

On the other hand, the storage, interchange and manipulation of formatted text as a means of representing structured data is becoming more and more common in any kind of application, from textual data base and digital libraries to web services and electronic commerce. A formatted text, and particularly those following XML standard, is becoming a model to follow to encode information with a simple or complex structure on one hand, and fixed or variable on the other. Although some time ago, XML was foreseen as a means to describe structured data, it was not until the recent boom of the "electronic bussiness" when it has shown its real potential to describe any kind of documents (invoices, bills, receipts, remunerations, etc) which are stored and interchanged in firm.

Although, the data managed in a firm is usually stored in relational data bases or data warehouse, it is important to have a digital copy, in XML format, of all the documents which have been produced or interchanged with time. The reason is that a structured information retrieval system can provided random access to those formatted documents and it will be easy

to search, show and browse. As a value-added, is desirable that size of this repository is the smallest as possible.

So there are different levels of structure in documents, on one hand, documents which represent books, papers, news, etc, have a simple structure and the contents of their structural elements are composed of written text in natural language. The size of these structural elements can be large. On the other hand, documents stored and interchanged in a firm are likely to have a more complex structure and their contents are much smaller and composed of names, references, quantities, short descriptions, etc... From these two types of structured documents can be defined:

1. Semistructured documents: those having tags (structure elements) which are used to organized in a semantic way the contents of the documents. The structure of these documents follow a DTD (Document Type Definition) and therefore its structure is variable, that is to say, two documents following the same DTD can have a different structure.
2. Structured documents: those having a fixed structure containing tags which semantic meaning goes beyond the proper tag's text, since they are an ASCII r epresentation of data types.

This thesis covers the following objectives, which are aimed at designing an information retrieval system which allows searching by content and structure keeping the documents compressed:

1. In a theoretical framework of an IRS which handles documents of semistructured text, it is interesting not only to retrieve documents but also the elements of structure which compose the document. Therefore it is necessary to define and design an inverted index which takes into account structural elements of the semistructured documents as atomic entities, so they can be retrieved as an answer to the searches by content and/or structure. With this index we can answer questions like: "Which documents have been written by author X?", "Which documents have an abstract?", or "In which theorems we can find the words X and Y?", supposing that "author", "abstract" and "theorem" are represented as structural elements in a DTD. Moreover, we have to remember the hierarchy of those documents since an atomic entity (structural element) which appears at level  $n$  may be made up of a set of atomic entities (structural elements) of level  $n + 1$ .
2. On the other hand different methods of compression which take into account the structure when compressing the documents will be proposed, studied and designed. It is supposed that these methods will obtained better compression rates because the structure of the documents is organized bearing in mind semantic factors and therefore

the distribution of the symbols contained in the structural elements of a type is different from the distribution of the symbols in the structural elements of other type. Likewise, considering the structure in the compression process, makes easy the possibility of solving queries by content and structure of the compressed text.

## 4 Contributions of the Thesis

This thesis proposes a general compression model thought to compress semistructured documents, the structural contexts model. This method is based on the semantic content of the different types of structure elements and it can be considered as an extension of the separated alphabet model [BSTW86]. It uses a specific model to model and code each set of types of structure elements with similar semantic meaning. In a first stage, a semiadaptive implementation has been developed, using a semiadaptive Huffman code, to use it in textual databases of compressed documents [ANF03a, ANF03b] and in a second stage a adaptive version has been developed, using PPM+, to use it in the storage and transmission of the documents [AFN04]. the compression rates obtained in each implementation are much better than those obtained respectively in the semiadaptive and adaptive models related in the section State of the Art.

Although the structural contexts model can be applied on structured documents, a compression technique inspired by the Lempel-Ziv schema is proposed to handle and compress the structure in a more accurate way than the general model [ANF04]. The main element of the schema is the LZCS Transformation, which is in charge of replacing the complete structural elements by a reference of their first occurrence. The reference is represented by a special tag which contains the position of the occurrence in the transformed text. The advantage of the structural context model is that the result is plain text with references which makes the display, browse and search easy.

An inverted index addressing structural elements is proposed to recover documents with or without compression by content and structure [AFVV02a, AFVV02b]. In this kind of index, the positions in the occurrence lists do not indicate positions in files but references in a table. This table is call structure control table and each line holds the information needed to resolve the search by content and structure. The proposal's rates are between those of inverted indexes addressing words and documents.

Although this research (structural context schema and inverted index with addressing structural elements) is focused to the field of IRS and textual database with compressed documents, they can be used not only in these fields but in others like digital libraries [AFV<sup>+</sup>02].

## 5 Structure of the Thesis

The material of the thesis has been distributed in two different parts: the “state of the art”, where some related works which have been the base of this research are commented. And “Contributions”, where the researches and proposals which constitute the thesis are detailed.

The section called “State of the Art” relates the elementary concepts connected with the IRS which keeps their documents compressed. First of all, in Chapter 2 the most general theoretical concepts related with data compression will be detailed. These concepts will be used in the statement of this work. Besides different classifications of the most usual compression techniques will be enunciated. In this Chapter it also can be found a demonstration of the count theorem, which is very important but very unusual in the related literature.

Chapter 3 details the description and analysis of the models, methods and techniques of text compression (it is also known as lossless compression) which will be the starting point of the proposals. The use of certain lossless compression algorithm allows searching on compressed text, in accordance with it, Chapter 4 explains briefly classic techniques used for searching patterns in a text without considering the structure of the document at first. In this Chapter, we can also find different standard techniques of indexing text, as well as those used to compress the index and they will be useful to understand the inverted index addressing structural elements and its compression. To finish this section, Chapter 5 briefly comments on basic aspects and concepts which are present in an IRS, it also comments on the different models which intervene between the user’s need of information and the answer of it. In the same way, it includes an exhaustive revision of the modern techniques which are used in IRS which work with semistructured documents.

The contribution section starts in Chapter 6. This chapter enunciates the proposal of a generic compression model which bears in mind and takes into account the structure of semistructured documents. This model is called “structural contexts model” and it has been thought up to compress semistructured documents in general. Given that the model is generic, there are different implementations; in this chapter we can find both an adaptive and a semiadaptive implementation. In both cases the ratios of compression are very competitive. In the semiadaptive version random searches and accesses can be performed, this is impossible in the adaptive one. However the ratios of compression are much better in the second case, showing clearly the inverted proportion between those functionalities and the ratio of compression.

On the other hand, and although the structured context model can be used to compress structured documents, a compression technique based on the idea of Lempel and Ziv has been design to compress them. Chapter 7 details this technique, which is due to the conversion of structured documents

before compressing them using any standard technique. The main idea of the conversion is to replace the substructures which are repeated by a pointer to the position of the first one. The result of the conversion is plain text, so it can be easily displayed, browsed and searched. It can be added to the compression's rates are really good.

Although in the literature there are some proposals of indexing schemes which take into account the structure to resolve queries by content and structure, Chapter 8 sets out a proposal called "inverted index addressing structural elements" which makes easier to resolve in a efficient way the said queries. In this chapter it is also defined a new model of information retrieval for documents with structure so that it is possible to solve queries by content and structure weighting each structural element by the density of the query terms and by the proximity between them.

Finally, Chapter 9 comments on the most significant results obtained during the development of the proposals enunciated in this work to achieve the initial objectives and it also outlines future work to continue with the work started in this thesis.

## 6 Conclusions and Future Work

In the framework of this thesis, the semantic aspect of the structure has been considered in the process of compression, indexing and retrieval of documents with structure. One of the main objectives has been to study and to take advantage of the structure of the documents in the processes previously mentioned. An IRS which handles compressed documents with structure can take advantage of their structure to improve the compression ratios so at the same time it allows to resolve efficiently searches by content and structure and to show the result in an appropriated way. To improve the compression ratios, a generic model which allows compressing semistructured documents based on the semantic aspect of the structure has been proposed. The key idea is to consider that the frequencies of the words in the text inside the same structural element must follow the same distribution. The model is called structural contexts model (SCM) and it encodes in the same model the texts which are in each type of structural element. In its semiadaptive version, called SCMWBH, words based Huffman encoders have been used. This idea has been improved by the addition of a heuristic, which allows grouping different types of structural elements, so each group is encoded with an independent specific model. On the other hand, the model's impact in the efficiency of the process is insignificant; in fact, it is similar to de efficiency in the retrieval of compressed documents.

It has been demonstrated, that nowadays, the compression's ratios of this idea are more than 10% better than those of the basic technique (separated alphabet model). On the other hand, the prototype has been compared to

the systems mentioned in the section State of the Art and the result shows that the prototype has the best ratio of compression using medium and large collections (which are the most interesting in the field of textual databases: 40 Megabytes or more).

In very large collections, the difference between the prototype and the best system is under 7.2% but that system does not allow the direct access to the compressed text and the prototype does.

An adaptive version has been proposed too, the SCMPPM, and it combines the generic technique, called SCM and the PPMD+ model to store and transmit semistructural documents.

It has been proved that this version improves the compression in more than 1% in comparison with a single PPMD+ model and in a 25% in comparison with the compressors mentioned in the State of the art section.

On the other hand, it also has been presented a method called LZCS, it is a compression schema based on the Lempel-Ziv schema, designed to compress structural documents. The main idea of LZCS is to substitute the complete substructures by a reference to a previous occurrence of them, so it can eliminate the redundancy which is introduced by the structure in collections of documents. In general terms the LZCS transformation has the following advantages:

1. Very good compression ratios, which improve the ones obtained by classical methods and by the compression methods which consider the structure.
2. Ease in the process of browsing, displaying and random access in compressed collections.
3. Compression and decompression in just one step.
4. Speed in the compression and decompression process.

In the experiments done, only XMLPPM compresses better than LZCS but as it has been said, XMLPPM cannot make random accesses to a particular document because it is adaptive and it needs to decompress all the documents which are before the desired document. Due to this fact, the use of XMLPPM in the field of compressed textual databases is rejected.

Perhaps a problem which can be attributed to LZCS is the efficiency in the phase of compression, which has a square complexity, using the definition proposed. To solve it, it can be designed an algorithm of compression with linear complexity using a hash schema.

In same scenes documents can be added to the collection, but they can neither be modified nor deleted. LZCS can be adapted easily to allow the insertion of new documents, but it is a bit harder to allow the modification and the deletion of documents. In the same way, it is interesting to design indexing schemas to perform fast searches for documents which contains

same given terms or substructures, bearing in mind that the collection is compressed.

The classical process of information retrieval considers documents as the minimum information unit which is indexed and retrieved as a whole. The modern development of the design and storage of documents has introduced from same time ago, more elaborated document representations; standards like SGML, HTML and currently XML, are a great contribution to this field. These standards are the fundamental cause of the current evolution towards the modern electronic documents. In this context, the retrieval of structural documents refers to the indexing and retrieval of information following a particular structure of documents. This means that documents are no longer considered as the smallest entities, but as set of related objects which can be retrieved separately, given a query it can be retrieved the set of document's components which are more relevant for that query.

The information retrieval for XML documents is a field of work which is getting more and more important. The use of the structure and content of these documents in the retrieval phase is an ever-present theme in all the works which are being developed in the subject. In that sense, it has been proposed an indexing schema which is able to retrieve structural elements, called "inverted index addressing structural elements", its features are between those from an inverted index addressing words and those from an inverted index addressing documents.

It also has been developed a information retrieval model which takes as its starting point two ideas which have been demonstrated to be valid during the weighing of the structural elements: the density of the terms in the query and its proximity. The proposed model takes into account both ideas, however about the proximity of the query terms, it only weighs if the terms appear or not in the same structural elements and the proximity of these terms inside the structural elements. With this approach, it is obtained a ranking of the nodes which adapts neither to a pure boolean approach nor to a vectorial one, because in same situations, nodes with some terms missing can be found in a better position than others which contain all the query terms, depending on the quantity and the proportion of the query terms present in that node. The idea pursued is to retrieve the elements which can be more interesting for the user from the point of view of the information's concentration, that is to say, the user is interesting in retrieving those documents (or parts of them) which have the asked information (or part of it) in a more concise<sup>2</sup> form.

---

<sup>2</sup>That is to say, expressing much in few words; brief and to the point.

## 6.1 Directions for Future Work

As a result of work made in this thesis, some questions and themes which seem interesting to go more deeply into them have been identified. On one hand, SCMWBH is a prototype with a basic implementation which can be improved to make it more competitive. For example, the technique used to fuse together the models can be refined because it has been confirmed that in general, the predictions made are approximately a 98-99% of the real size, so an average value can be added to each prediction, allowing a more accurately approach to the real size.

Relating to the application of the model to text blocks, the result has not been very promising, nevertheless its effect can be proved with other type of collections in which the distribution of the words in the text of each structural element change in different parts of them. With regard to the study of the method itself, the relation between the type and the density of the structure must be studied in-depth and in connection with the improvements in the method, as its success is based in a semantic assumption and it could be interesting to check how it works with other collections of text.

In relation to the adaptive version of SCM, at present there are works on progress about the use of words and delimiters as the base of the source alphabet of input symbols for the model PPM, as words represent the entropy of the text better than characters [BCW90]. For example, a semiadaptive Huffman encoder which considers characters as symbols of the source alphabet will obtain, on average, a compressed file whose size is about 60% of the size of the original file when processing text written in natural language. A semiadaptive Huffman encoder which considers words as symbols of the source alphabet will obtain, on average, a compress file whose size is about 25% of the original size [ZMNBY00]. Another example is the WLZW (Ziv-Lempel on words) algorithm [BSTW86, DPS99].

It can be emphasized that SCMPPM is equivalent to have an extra context made up by the last element of the structure processed, which precedes the first character of the current context. From this point of view, several generalization can be made, for example using more than one structural context (for instance, the last two elements of the structure which contain the current text) and insert the structural context with the character in a different order to the current one (first of all the structural context and the context of the character).

On the other hand, the work with LZ compression schemas has planned the idea of studying the possibility of using it together with the techniques of pattern matching with errors. The main idea is to use an LZ77 schema to search the string in the sliding window, this string can have one or more differences with regard to the string of the current position, so it can be performed a greater substitution but it is necessary to encode the positions which produce the errors and their value to recover the original string in

the decoding process. It is necessary to check that this idea obtains improvements in relation to the basic schema, in addition, heuristics which allow selecting and to encode the differences between the strings must be established.

Finally, with regard to the indexing and retrieval process, the use of the inverted index addressing structural elements suggests the possibility of performing searches by proximity taking into account the semantic aspect of the texts. In relation to the proposed model, it can be made a more in-depth analysis and more queries on bigger standard collections, like INEX.

It can be interesting to adapt the model to obtain a list of the “best entry points” of the documents instead of their general classification. This possibility emerges from the study of the factors which the model considers when obtaining the weights of the nodes on one hand and the results on the other. Let us remember that the “best entry points” are those parts or places in the documents, returned as the answer to a query, which are suggested to the user as a starting point of reading in the document. The fact that the model gives priority over the parts of the documents where a bigger concentration of query terms exists leads us to think that the application of that model for that purpose gets satisfactory results.

## References

- [AFN04] J. Adiego, P. de la Fuente, and G. Navarro. Merging prediction by partial matching with structural contexts model. In *Data Compression Conference (DCC'04), Snowbird, Utah, USA*, page 522. IEEE Computer Society TCC, March 2004.
- [AFV<sup>+</sup>02] J. Adiego, P. de la Fuente, J. Vegas, M. Villarroel, and A. Pedrero. Bibliotecas digitales con documentos comprimidos: una arquitectura. In *Terceras Jornadas de Bibliotecas Digitales (JBIDI 2002), El Escorial, Madrid, España*, pages 133–142, Noviembre 2002.
- [AFVV02a] J. Adiego, P. de la Fuente, J. Vegas, and M. Villarroel. Una técnica para compresión y acceso de documentos estructurados. *Novática, Número monográfico: Recuperación de información y la Web*, 157:34–40, Mayo/Junio 2002.
- [AFVV02b] J. Adiego, P. de la Fuente, J. Vegas, and M. A. Villarroel. System for compressing and retrieving structured documents. *Upgrade - The European Online Magazine for the IT Professional - Information Retrieval and the Web*, III(3):62–69, June 2002.

- [ANF03a] J. Adiego, G. Navarro, and P. de la Fuente. Compressing semistructured text databases. In *25th European Conference on Information Retrieval Research (ECIR'03), Pisa, Italia*, pages 153–167. Springer-Verlag, LNCS 2633, April 2003.
- [ANF03b] J. Adiego, G. Navarro, and P. de la Fuente. SCM: Structural contexts model for improving compression in semistructured text. In *10th International Symposium on String Processing and Information Retrieval (SPIRE 2003), Manaus, Brazil*, pages 153–167. Springer-Verlag, LNCS 2857, October 2003.
- [ANF04] J. Adiego, G. Navarro, and P. de la Fuente. Lempel-ziv compression of structured text. In *Data Compression Conference (DCC'04), Snowbird, Utah, USA*, pages 112–121. IEEE Computer Society TCC, March 2004.
- [BCW90] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, N.J., 1990.
- [BPSM00] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0*. W3C Consortium, second edition, 2000. <http://www.w3.org/TR/REC-xml>.
- [Bra00] N. Bradley. *The XML Companion*. Addison-Wesley, second edition, 2000.
- [BSTW86] J. Bentley, D. Sleator, R. Tarjan, and V. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29:320–330, 1986.
- [BVNF98] R. Baeza-Yates, J. Vegas, G. Navarro, and P. de la Fuente. A model of visual query language for structured text. In *Proceedings of SPIRE'98*, pages 7–13. IEEE Computer Society, Sep 1998.
- [BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, may 1999.
- [Che01] J. Cheney. Compressing XML with multiplexed hierarchical PPM models. In *Proc. Data Compression Conference (DCC 2001)*, pages 163–, 2001.
- [DPS99] J. Dvorský, J. Pokorný, and V. Snásel. Word-based compression methods and indexing for text retrieval systems. In *Advances in Databases and Information Systems, Third East European Conference, ADBIS'99, Maribor, Slovenia, September 13-16, 1999, Proceedings*, volume 1691 of *Lecture Notes in Computer Science*, pages 75–84. Springer, 1999.

- [Eli75] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, IT-21(2):194–203, 1975.
- [HMQS96] G. Haider, C. Magnusson-Sjöberg, G. Quirchmayr, and V. Sebald. The comparative part of the corpus legis project - using SGML for intelligent information retrieval of legal documents. In *EXPERSYS-96, Artificial Intelligence Applications*, pages 181–186, 1996.
- [Huf52] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. Inst. Radio Engineers*, 40(9):1098–1101, 1952.
- [ISO86] ISO 8879:1986. Standard Generalized Markup Language (SGML). *Information Processing - Text and Office System*, Oct 1986.
- [LS00] H. Liefke and D. Suci. XMill: an efficient compressor for XML data. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153–164, 2000.
- [LYYB96] Y. K. Lee, S.-J. Yoo, K. Yoon, and P. B. Berra. Index structures for structured documents. In *ACM First International Conference on Digital Libraries*, pages 91–99. ACM, 1996.
- [MNZ97] E. S. de Moura, G. Navarro, and N. Ziviani. Indexing compressed text. In Carleton University Press International Informatics Series, editor, *Proceedings of the Fourth South American Workshop on String Processing*, pages 95–111, 1997.
- [MNZB00] E. S. de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, 2000.
- [Mof89] A. Moffat. Word-based text compression. *Software - Practice and Experience*, 19(2):185–198, 1989.
- [MT97] A. Moffat and A. Turpin. On the implementation of minimum-redundancy prefix codes. *IEEE Transactions on Communications*, 45(10):1200–1207, 1997.
- [MW01] A. Moffat and R. Wan. RE-store: A system for compressing, browsing and searching large documents. In *Proc. 8th Intl. Symp. on String Processing and Information Retrieval (SPIRE 2001)*, pages 162–174, 2001.

- [MZ00] E. S. de Moura and N. Ziviani. Construção eficiente de índices para bases de dados textuais. In *SBBD'2000 - XV Simpósio Brasileiro de Banco de Dados. João Pessoa. Paraíba*, Oct 2000.
- [NBY97] G. Navarro and R. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *Information Systems*, 15(4):400–435, 1997.
- [NMN<sup>+</sup>00] G. Navarro, E. S. de Moura, M. S. Neubert, N. Ziviani, and R. Baeza-Yates. Adding compression to block addressing inverted indexes. *Information Retrieval*, 3(1):49–77, 2000.
- [TH02] P. Tolani and J. R. Haritsa. XGRIND: A query-friendly XML compressor. In *ICDE*, 2002.
- [VFC02] J. Vegas, P. de la Fuente, and F. Crestani. A graphical user interface for structured document retrieval. In *Advances in Information Retrieval, 24th BCS-IRSG European Colloquium on IR Research Proceedings*, volume 2291 of *Lecture Notes in Computer Science*. Springer, March 2002.
- [WMB99] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, Inc., second edition, 1999.
- [ZL77] J. Ziv and A. Lempel. An universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, 1977.
- [ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24(5):530–536, 1978.
- [ZM95] J. Zobel and A. Moffat. Adding compression to a full-text retrieval system. *Software - Practice and Experience*, 25(8):891–903, 1995.
- [ZMNBY00] N. Ziviani, E. S. de Moura, G. Navarro, and R. Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, November 2000.