Using Attribute Grammars to Uniformly Represent Structured Documents - Application to Information Retrieval

Alda Lopes Gançarski
Pierre et Marie Curie University, Laboratoire d'Informatique de Paris 6, Paris, France

Abstract

This paper presents an ongoing work to uniformly represent structured documents by mean of Attribute Grammars (AG). Each document corresponds to a syntactic tree with nodes decorated with sets of attributes. The values of these attributes correspond to characteristics which specify the semantics of both the textual content and the structural elements. We show how to use this representation for the Information Retrieval (IR) task from collections of structured documents. We give a brief global overview of the proposed *DASTIR* system, describing the specification of the syntactic and the semantic parts of the AG generated to give the desired response to a structural query.

Introduction

It has been recently a tremendous growth of the specification of structured textual information using the standards *Standard Generalised Mark-up Language* (SGML), *Hypertext Mark-up Language* (HTML) and *eXtensible Markup Language* (XML). Initially, the purpose of the use of marks in documents was to show how texts should be printed or displayed. We can say that the process of marking a text is a way to make its interpretation explicit. When a collection of documents share the kind of information they deal with, it is natural to think about describing that information in the same way for all of them. For example, business letters have almost all the same structure, so it is desirable that this structure would be described in a standard way. This ideas where at the origins of a standard for specifying *markup* languages, the SGML, as a format of exchanging documents. After that, the HTML was developed as an SGML application to show the documents in the world wide web. This standard makes use of mainly presentation marks to describe how the textual parts must be displayed by the browser, without taking into account the real structure of a document. More recently, XML appeared as a simplification of SGML to be able to exchange documents in the web. XML documents have a description richer than the simple but limited one provided by HTML. The specification of the structural elements and their hierarchical relations for a certain type of documents is made through the *Document Type Definition* (DTD).

Our work is based on the idea that we can take advantage of the markup information to represent structured documents in a standard way in order to extract results from them, whatever the system application. A natural way of representing the structure and the semantics of structured documents is by means of a AG ([RLH98], [NB98]). Not only it is a wide known and rather simple concept, but it also allows to uniformly represent heterogeneous sources of structured information, avoiding to develop new intermediate specifications or languages for specialised tasks. We argue that the AG formalism is powerful enough to express multiple operations over documents collections.

A AG consists of a context independent grammar extended by a set of attributes (and rules for their calculation) which specify the semantics of the analysed texts. If necessary, it also allows to impose contextual conditions to productions, based on attribute values. The result of the syntactic and semantic analysis of a text is an abstract syntax tree decorated with the attribute values (DAST). When an attribute value is calculated by the production where the respective symbol is derived (i.e. it is on the left-hand side of the production) it is called synthesised attribute. Otherwise, the attribute is called inherited. Synthesised attributes make the propagation of the semantic information from the leaves up to the root of the syntax tree, while the inherited ones do the same but from the root down to the leaves or between sibling nodes.

In this work, we focus on Information Retrieval (IR) using a simple format of structural queries. IR consists of retrieving the relevant documents to a query, while returning as few as possible of non-relevant documents. Moreover, the resulting documents should be ranked by their relevance to the query. Simple textual representations, like "bag-of-words", are used in IR to filter or classify the information via statistical techniques. One possible representation is a vector in which the components store a measure of how representative is each term to the meaning of the text. This measure is based on both the frequency of the term in a document and the frequency of the term in all the documents. With the emergence of structured documents, the IR evolved in two directions: (1) retrieving documents taking into account the structural parts relevance ([Wil94], [Hea94]) and (2) enriching the query formats with structural information to retrieve certain parts of documents ([NBY95], [KM93]). Recent works tried to establish some form of relevance ranking in the results ([Lal00], [WFC99], [HTK00], [SN00]), but it is still an opened research area.

The *DASTIR* System

Our system is composed of different modules, as shown on the Figure 1. Assuming the existence of the DTD (if not, it can be inferred [FX93]), the AG Generator module will automatically generate a specific AG, as we will show in the next section. The AG Analyser makes a syntactic and semantic analysis of an input document to create the corresponding DAST. The textual contents and the structural information are used as arguments of various external functions which compute the attribute values (for example, a function can compute the size of a text). The query is a pair *(element, textual restriction*) which expresses the type of element to return and a textual restriction over it (expressed as a natural language query). The AG analyser assign to each occurrence of the specified element a measure of the relevance to the query. For that, it takes into account the relevance of the textual information that the element contains, as well as its structural characteristics (for example the tag name). The result of analysing one document is the ranked list of the relevant occurrences. After analysing all the documents, the resulting lists can be merged to have a final ranked list of all the relevant element occurrences. This result can be given to the user in several ways; for example, each occurrence may have a pointer to the document it belongs to.

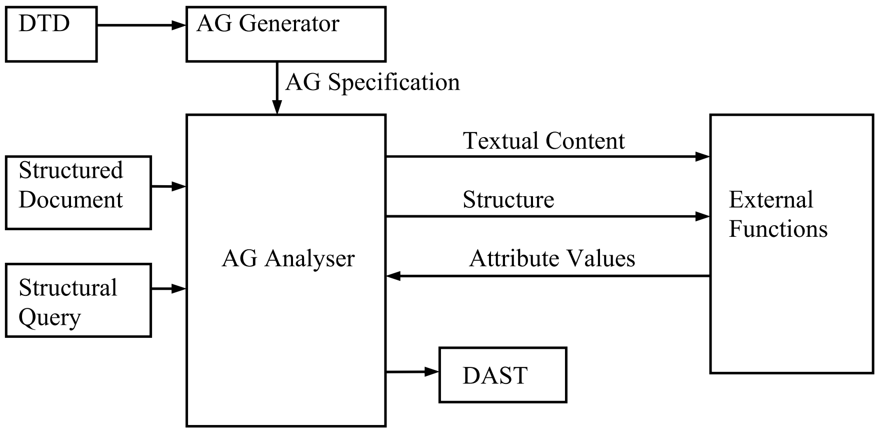

Figure 1 - The *DASTIR* system.

The AG Generator

To represent a structured document by a DAST, we need to specify both the syntactic and the semantic components of the AG. The syntactic component of the AG corresponds to the structure of the document. The productions can be automatically constructed from the element declarations using a pre-defined set of generic mapping rules. Each rule corresponds to a type of element declaration or a type of operator in the regular expressions of the element contents. Due to space limitations, we omit here those rules, but the reader can refer to [RLH98] for a similar approach.

The textual content is represented by a non-terminal symbol, *Text*, which derives in a terminal symbol that corresponds to a string. The set of attributes for the IR task is:

- *identifier*: the unique identifier for the element occurrence;
- *value*: the intrinsic value of TEXT, not needing to be calculated;

- *elem_att*: set of element characteristics, such as its parent, the number of children, the children names, the path from the root[1];
- *text_att*: set of textual characteristics, such as the size and the vector representation;
- *relevance*: the relevance of an element to the query, taking into account the textual and the structural characteristics;
- *select*: binary value to detect the elements of the type specified in the query;
- *old_rank_list*: ranked list of relevant elements inherited from the partial DAST constructed until the present element; it is initialised with an empty list in the production of the root symbol;
- *new_rank_list*: ranked list of relevant elements updated with the information coming from the partial DAST with the present symbol as root.

An Example

We show now an example composed by a DTD which describes letters (in Figure 2), a corresponding instance (in Figure 3), a query and the final DAST representation of the instance (in Figure 4).

As specified in the DTD, each letter is composed by a textual element called *Head* followed by the element *Message*. A *Message* is formed by a list of paragraphs, each one defined by the textual element *Paragraph*. The letter given as example of this type of documents has a *Head* and a *Message* composed by two paragraphs. The query asks for paragraphs talking about vacations (*query = (Paragraph, "vacations")*). Based on the DTD, the document and the query, the *DASTIR* system represents the document by a DAST where the answer to the question is given.

In the DAST, the synthesised attributes appear on the right side of the derivation arrows and the inherited ones, i.e. the ones that depend on attributes coming from the ascendants, on the left side. For simplicity, the values of the attributes *elem_att* and *text_att* are omitted here. In this example, the *relevance* attribute has a simplified calculation function: it is the frequency of the textual restrictions in the textual content, here the frequency of the term *"vacations"*. Each symbol corresponding to an element is assigned a unique natural number identifier. The *select* attribute has always the value 0, except for the two occurrences of the desired element (*Paragraph*). The old ranked list of occurrences of the desired element (*old_rank_list*) for the symbol *Letter* is an empty list since there is no partial DAST constructed until that node. Then, it is inherited by *Head* (for clarity, the evolution of the list is indicated with dotted arrows). The corresponding updated list (*new_rank_list*) remains empty because *Head* is not the desired element. This list is inherited by *Message* and then by the first *Paragraph* in the *old_rank_list* attribute. The attribute *select* of this paragraph has the value 1. Thus, a relevance value of 1 is stored in *relevance*, which means that there is one occurrence of the term *Vacations*. The new ranked list becomes a single list with this paragraph's identifier *(4)*. The following *Paragraph* has a relevance value of 2; consequently, the head of the ranked list becomes the identifier of this paragraph *(5, 4)*. This list is now passed up to *Message* and then to *Letter*. The final result of the IR process over the instance is the *new_rank_list* attribute of the root symbol, i.e., *(5, 4)*, which means that the *Paragraph* corresponding to the identifier *5* ("*I am in vacations! This year I plan to visit all Europe! I hope my vacations will be good! Bye!"*) is the most relevant to the query, followed by the *Paragraph* corresponding to the identifier *4* (*"Hello! Are you already in vacations?"*).

```
< !DOCTYPE   Letter [
< !ELEMENT   Letter  (Head, Message) >
< !ELEMENT   Head  (#PCDATA) >
< !ELEMENT   Message (Paragraph)+ >
< !ELEMENT  Paragraph  (#PCDATA)>   ]>
```

Figure 2 : The DTD for documents of type *Letter*.

```
<Letter>
<Head>14 Nov 2000, Dear Marianne: </Head>
<Message>
<Paragraph>Hello! Are you already in vacations?</Paragraph>
<Paragraph>I am in vacations! This year I plan to visit all Europe! I hope my
vacations will be good! Bye!</Paragraph>
```

[1] For HTML documents, this characteristics can be specialised to give, for example, more weight to elements like *Title* or *Head*.

</Message>
</Letter>

Figure 3 : A XML document of type *Letter*.

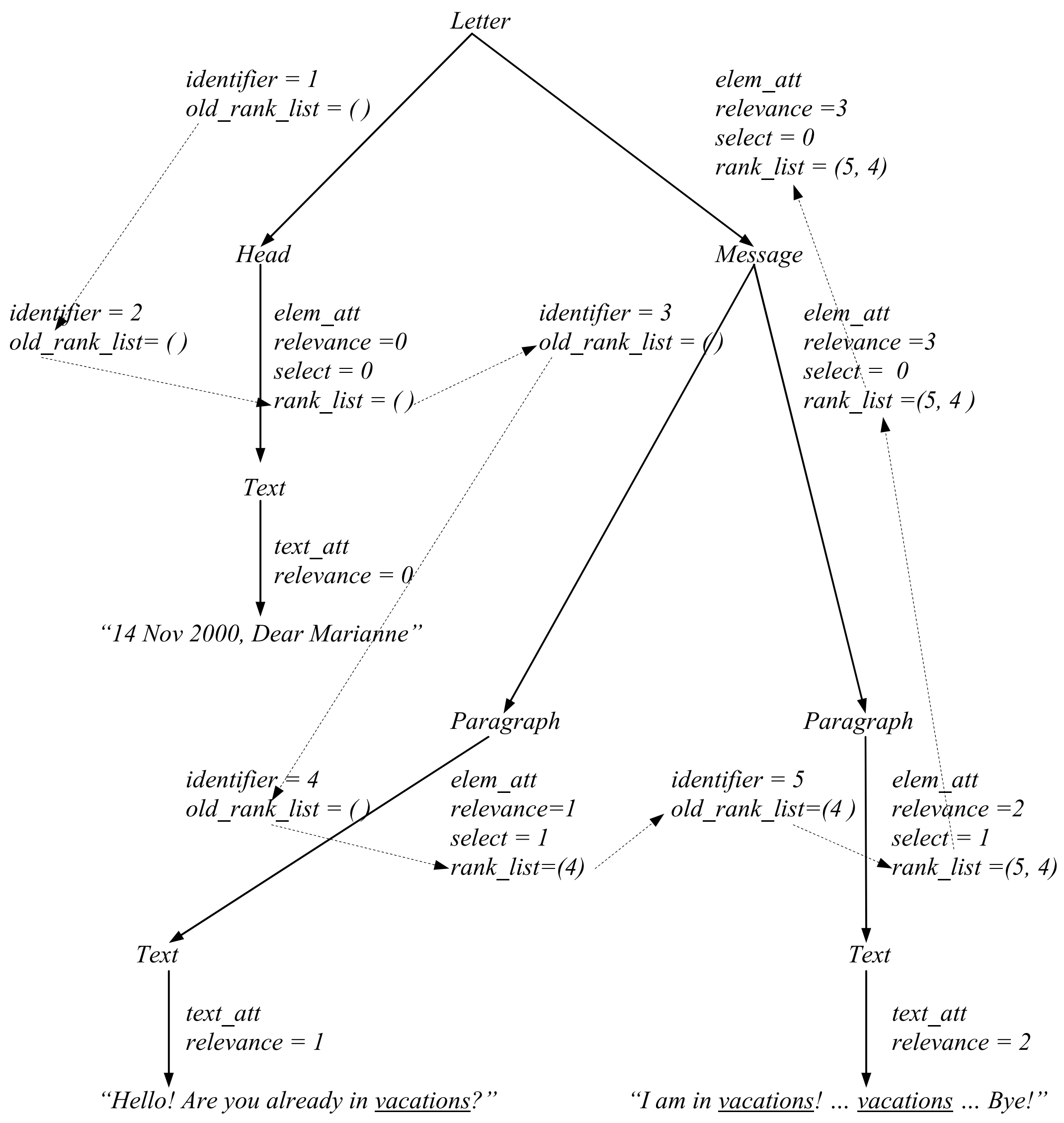

Figure 4 : The final DAST representation of the example XML document.

Conclusions

We have presented *DASTIR*, a system to make IR over collections of structured documents represented in DASTs. This simple and effective representation can be shared by multiple applications by just defining the dedicated external functions to achieve the specific goals. This approach can be extended to collections of other document formats with some internal structuring scheme, such as PDF or latex. In this case, it is necessary to develop the rules to map the marks of the documents in productions of the AG. These rules will be used in the AG generator module similar to the one used for XML in this paper.

As future works, we intend to extend the AG specification in order to include attributes, entities, hyperlinks and other multimedia information in the automatic generation of the AG. We believe that their semantic can improve the IR task. When the documents are enriched with metadata, it can be used too to improve the semantic specification. Also, it is interesting to evaluate the possibility of extending the system to accept more flexible structured queries.

Financial Support

This work is financially supported by the *"Sub-Programa Ciência e Tecnologia do 2° Quadro Comunitário de Apoio"* of the Portuguese Foundation for Science and Technology.

References

[RLH98] J. Ramalho, A. Lopes and P. Henriques, *Generating SGML specific editors: from DTDs to attribute grammars*, Mark-up Technologies'98, Chicago - USA, 1998.
[NB98] F. Neven and J. Bussche, *Expressiveness of Structured Document Query Languages Based on Attribute Grammar*, ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pages 11-17, ACM press, 1998.
[Wil94] R. Wilkinson, *Effective retrieval of structured documents*, in ACM SIGIR'94, 1994.
[Hea94] M. Hearst, *Multi-paragraph segmentation of expository text,* 23nd Annual Meeting of the Association for Computational Linguistics, pages 9-16, New Mexico State University, Las Cruces, New Mexico, 1994.
[NBY95] G. Navarro and R. Baeza-Yates, *A language for queries on structure and contents of textual databases*, ACM SIGIR'95, 1995.
[KM93] P. Kilpelainen and H. Mannila, *Retrieval from hierarchical texts by partial patterns*, ACM SIGIR'93, 1993.
[Lal97] M. Lalmas, *A dempster-shafer theory of evidence applied to structured documents: Modelling uncertainty*, Research and Development in Information Retrieval, pages 110-118, 1997.
[WFC99] J. E. Wolff, H. Florke and A. B. Cremers, *Xpres: a ranking approach to retrieval on structured documents*, Technical report, University of Bonn, Romerstr. 2164, D-53117, Germany, 1999.
[HTK00] Y. Hayashi, J. Tomota and G. Kikui, *Searching text-rich XML documents with relevance ranking*, ACM SIGIR 2000 Workshop on XML and Information Retrieval, Athens, Greece, 2000.
[SN00] T. Schlieder and F. Naumann, *Approximate tree embeeding for querying XML data*, ACM SIGIR 2000 Workshop on XML and Information Retrieval, Athens, Greece, 2000.
[FX93] P. Fankhauser and Y. Xu, *MarkItUp! An incremental approach to document structure recognition*, pages 447-456, 1993.