Auto-tagging of Text Documents into XML

Shazia Akhtar[1], Ronan G. Reilly[2], and John Dunnion[1]

[1] Smart Media Institute, Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland
{Shazia.Akhtar,John.Dunnion}@ucd.ie
[2] Department of Computer Science,
National University of Ireland Maynooth,
Maynooth, Co Kildare Ireland
Ronan.Reilly@may.ie

Abstract. In this paper we present a novel system which automatically converts text documents into XML by extracting information from previously tagged XML documents. The system uses the Self-Organizing Map (SOM) learning algorithm to arrange tagged documents on a two-dimensional map such that nearby locations contain similar documents. It then employs the inductive learning algorithm C5.0 to automatically extract and apply auto-tagging rules from the nearest SOM neighbours of an untagged document. The system is designed to be adaptive, so that once a document is tagged in XML, it learns from its errors in order to improve accuracy. The automatically tagged documents can be categorized on the SOM, further improving the map's resolution. Various experiments were carried out on our system, using documents from a number of different domains. The results show that our approach performs well with impressive accuracy.

1 Introduction

The extraordinary growth of information resources has created vast and complex repositories of data. Such large amounts of data require the development of new procedures for storage and management. In addition, the need for efficient and effective search for specific information in growing repositories of data also requires new paradigms for data organization. The recent acceptance of XML as an emerging standard markup language has provided a solution for effective management and retrieval of large and highly complex data repositories. The idea behind XML markup (tagging) is to structure raw data, including natural language texts, with descriptive element tags. XML is not a set of tags itself: it provides a standard system for browsers and other applications to recognize the data in a tag. By using XML as a standard markup language, search engines can use XML tags to exploit the logical structure of documents, which should improve search results, avoid irrelevant searches and provide more precise information. However, despite the benefits provided by XML, we still do not have large collections of XML documents.

Manual tagging of a collection of text documents into XML is impractical because of the time, effort and expense required. For text documents to be efficiently and effectively converted into XML, the process of tagging must be automated. Currently auto-tagging is a significant challenge. Most systems that have been developed are limited to certain domains and require considerable human intervention. In addressing the problem of auto-tagging, we present a novel hybrid system that produces tagged document collections

© Springer-Verlag Berlin Heidelberg 2003

by using two machine learning techniques, namely the Self-Organizing Map (SOM) algorithm [1], [2] and the inductive learning algorithm C5.0 [3], [4]. The process of auto-tagging is based on the previously tagged valid XML documents to be used as training data by the system.(A valid XML document is one which is well-formed and which has been validated against a DTD).

2 Overall Approach

The hybrid architecture of our system combines the SOM and C5.0 algorithms to produce XML tagged documents.

The overall approach is shown in Fig. 1. Phase 1 of the hybrid system deals with the formation of a map of tagged documents using the SOM algorithm. Once a map has been formed, the system automatically extracts information from the SOM neighbours of an untagged document in phase 2. This information is extracted in the form of rules by using the inductive learning algorithm C5.0. These rules together with text segmentation heuristics derived from the set of tagged documents are used to markup the untagged text document into XML. These two phases of the system are currently implemented independently but will eventually be linked together to form an integrated hybrid system. Phase 2, which is the focus of this paper, is currently implemented as an independent auto-tagging process and is described in Sect. 3.

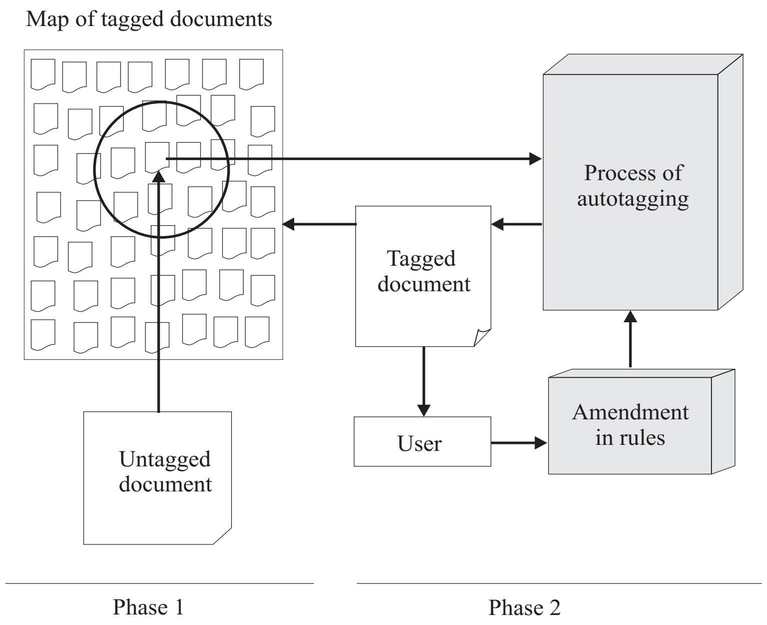

Fig. 1. Architecture of the hybrid system. *Phase 1* deals with the formation of a Self-Organizing Map. *Phase 2* deals with the auto-tagging of text documents into XML by using the inductive learning algorithm C5.0.

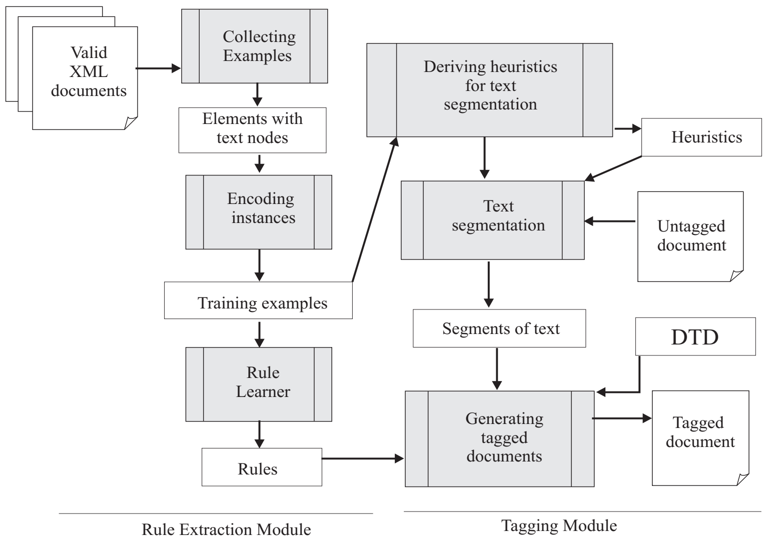

Fig. 2. The auto-tagging process.

3 The Auto-tagging Process

The auto-tagging process (Phase 2 of the hybrid system) is shown in Fig. 2. It has two main modules, a rule extraction module and a tagging module.

The rule extraction module learns rules from a collection of tagged documents using an inductive learning approach [5]. In this module, training examples are collected from a set of valid XML documents. These documents should be from a specific domain and their markup should be valid and comply with the rules of a single Document Type Definition (DTD). An XML document can be represented as a tree-like structure with a root element and other nested elements. Only elements having text are considered appropriate for our auto-tagging process. Each training instance corresponds to a leaf element containing text from the collection of tagged documents. The texts enclosed between the start and end tags of all occurrences of each element are encoded using a 6fixed-width feature vector. These encoded instances are used subsequently for learning the rules. Thirty-one features, such as word count, character count, etc., are used to encode the training instances. The system pre-classifies the encoded instances by the tag name of the element. These pre-classified encoded instances are used by the system to learn classifiers for the elements with that tag name. The learned classifiers are later used in the process of auto-tagging. We have used the C5.0 learning algorithm to learn classifiers. The advantages of this learning algorithm are that it is very fast, it is not sensitive to missing features and it is incremental. C5.0 is best suited for our system because it is not sensitive to missing features. Our system deals with documents from different domains, so some of the features are not relevant to the documents of all domains. Sets of rules are generated in a given domain from a collection of tagged documents and are used to markup the untagged text documents from the same domain.

The second module creates a tagged version of an untagged text document, which should be from the same domain as the documents used for learning the rules. The untagged document is segmented into pieces of text using a variety of heuristics. These heuristics are derived from the set of training examples. By applying the rules of the DTD, the rules extracted by using the C5.0 algorithm and the text segmentation heuristics, the hierarchical structure of the document is obtained and a tagged version of the text document is generated.

The tagged document produced by the system can be validated against the DTD by using any XML parser. However XML processors can only validate the syntax of an XML document. Since they cannot recognize the content of a document, a human expert is required to evaluate the accuracy of the auto-tagging process.

4 Experiments and Evaluation

For our experiments, we have used collections of documents from a number of different domains. These include letters from the MacGreevy Archive [6], [7], a database of employee records, Shakespearean plays [8], poems from the Early American Digital Archives [9] and scientific journal articles [10]. An example taken from *A Midsummer Night's Dream* automatically tagged by our system is shown in Fig. 3. The underlined text, with the start and end tags of the element *STAGEDIR*, is not tagged by our system. This represents an error made by our system.

All the documents sets used in our experiments except the scientific journal articles were tagged by applying the rules extracted by using the C5.0 algorithm, the text segmentation heuristics and the rules of the appropriate DTD. For the scientific journal articles we have used additional heuristics devised specifically for this domain. We hope that these heuristics can be used effectively for articles from most journals. The tagged journal articles used as training documents for our experiments were downloaded from the World Wide Web [10] along with the DTD (article.dtd) devised for these articles. From the same site, the HTML versions of articles were downloaded, converted to text files and automatically tagged into XML by our system. The XML DTD used for these tagged articles is complicated and requires the presence of another DTD (biblist.dtd) devised for references and bibliographies. For the auto-tagging of articles, currently we only consider those elements of DTD that descr ibe different sections of the article for example, title, author name, author affiliation, headings, paragraphs, references, etc. We have ignored the elements embedded in the text containing elements. These elements include the elements representing formatting or physical representation of different sections of the articles, e.g. `<b>`, `<i>` etc. Part of a scientific journal article automatically tagged by our system is shown in Fig. 4. Again, our system failed to tag the underlined text with start and end tag of *title* and *orgName*. Although the system makes some mistakes, it still works reasonably well with our domain-specific heuristics and automatically tags most of the sections of the journal articles.

We have used three performance measures to evaluate the performance of our system. These measures are:

– The percentage of elements correctly tagged by the system
– The percentage of elements incorrectly tagged by the system
– The percentage of elements not tagged by the system

```
…
<SCENE>
   <TITLE> SCENE I. Athens.  The palace of THESEUS.
   </TITLE>
   <STAGEDIR> Enter THESEUS, HIPPOLYTA,
   PHILOSTRATE, and Attendanrs</STAGEDIR>
   <SPEECH>
      <SPEAKER>THESEUS</SPEAKER>
      <LINE>Now, fair Hippolyta, our nuptial hour</LINE>
       <LINE>Draws on a pace; four happy days bring in</LINE>

      <LINE>Another moon: but, O, me thinks, how
      slow</LINE>
      <LINE>This old moon wanes!  she lingers my
      desires,</LINE>
      <LINE>Like to a step-dame or a dowager</LINE>
      <LINE>Long withering out a young man revenue. </LINE>
   </SPEECH>
   <SPEECH>
      <SPEAKER>HIPPOLYTA</SPEAKER>
      <LINE>Four days will quickly steep themselves in night;
      </LINE>
      <LINE>Four nights will quickly dream away the time;
      </LINE>
      <LINE>And then the moon, like to a silver bow</LINE>
      <LINE>New-bent in heaven, shall behold the night</LINE>
      <LINE>Of our solemnities</LINE>
      </SPEECH>
 …
```

Fig. 3. Part of a scene taken from *A Midsummer Night's Dream* automatically tagged by our system.

When describing the accuracy of our system, we use the first of these measures, i.e. the percentage of the tagged elements correctly determined by the system. Evaluation of the performance of our system for letters (from the MacGreevy Archive) demonstrates that it achieves an accuracy of 96%. For the Shakespearean plays, our system achieves 92% accuracy and for the poems taken from the Early American Digital Archives, it achieves 96% accuracy. For the scientific journal articles, the accur tagging process is 97%.

Conclusions

This paper describes a novel system which automatically tags the text documents into XML. The system uses the Self-Organizing Map (SOM) algorithm and the inductive learning algorithm C5.0 for the process of auto-tagging. The performance of our system has been evaluated in experiments with different datasets and the results indicate that our approach is promising. The functionality of our system makes it a useful tool for producing large tagged collections of documents.

```
 <?xml version="1.0"?>
<!DOCTYPE article SYSTEM article.dtd">
<article>
 <front>
     <docCiteAs>&nbsp;MRS Internet J. Nitride Semicond.
                    Res.3, 14.</docCiteAs>
    <cpyrt>&nbsp;1999 The Materials Research Society</cpyrt>
    <title>Surface Morphology of MBE-grown GaN on GaAs(001)
         as Function of the N/Ga-ratio</title>
    <Authors>
           <auth><pn>O. Zseb&ouml;k</pn></auth>
           <auth> <pn>J.V. Thordson</pn> </auth>
           <auth><pn>T.G. Andersson</pn></auth>
           <aff>
             <orgName>Chalmers University of Technology
                </orgName>
           </aff>
    </authors>
    <history><date>Tuesday, June 23, 1998</date></history>
    <history><date>Monday, August 24, 1998</date></history>
    <abstract>
           <p>Molecular beam epitaxy growth utilising an RF-plasma
nitrogen source was used to study surface reconstruction and surface
morphology of GaN on GaAs (001) at 580 &deg;C. While both the
nitrogen flow and plasma excitation power were constant, the grown
layers were characterised as a function of Ga-flux. In the initial growth
stage a (3x3) surface reconstruction was observed. This surface
  ....
```

Fig. 4. Part of a scientific journal article automatically tagged by our system.

Acknowledgements

The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged. The work was funded under grant PRP/00/INF/06.

References

1. Kohonen, T.: Exploration of very large databases by self-organizing maps. In Proceedings of ICNN'97, International Conference on Neural Networks. PL1-PL6. IEEE Service Center: Piscataway, NJ (1997a)
2. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Science (1997b)
3. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kauffmann, Los Altos, CA (1993)

4. Quinlan, J. R.: Data Mining Tools See5 and C5.0, [http://www.rulequest.com/see5-info.html] (2002)
5. Mitchell, T. M.: Machine Learning. McGraw-Hill, NewYork, NY (1997)
6. Schreibman, S.: The MacGreevy Archive. [http://www.ucd.ie/˜cosei/archive.html] (1998)
7. Schreibman, S.: The MacGreevy Archive. [http://jafferson.villiage.Virginia.edu/-macgreevy] (2000)
8. Bosak, J.: The Plays of Shakespeare in XML. [http://www.oasis-open.org/cover/bosakShakespeare200.html] (1999)
9. Schreibman, S.: Early American Digital Archives, Hosted by Maryland Institute of Technology [http://www.mith.umd.edu] (2003)
10. Hellman, E., Ephron, D., Poindexter, M.: Openly Informatics Inc. [http://www.openly.com/efirst] (1999-2000)