Automatic Document Markup using a Self-Organising Map

SHAZIA AKHTAR, JOHN DUNNION and RONAN G REILLY[†]
Department of Computer Science
University College Dublin
Belfield, Dublin 4
IRELAND

Abstract: In this paper we present a 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 an inductive learning algorithm 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 subsequently be categorized on the Self-Organizing Map, further improving the map's resolution. Our system has been evaluated on a number of different domains, giving promising results.

Keywords: Self-organising maps, automatic XML markup, machine learning.

1 Introduction

The extraordinary growth of information resources has created vast and complex repositories of data. To store and manage such large amounts of data requires the development of new procedures. In addition, the need for efficient and effective search for specific information in growing data repositories also requires new paradigms for data management and organization. The recent acceptance of XML as an emerging standard markup language has provided a solution for effective management and retrieval from large and highly complex data repositories. Using XML markup allow authors 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 listings of the information available. However, despite the number of benefits provided by XML, large collections of XML documents still do not exist. Manual tagging of text documents in XML is difficult and the time, effort and expense involved in producing a tagged collection is impractical. For text documents to be efficiently and effectively converted into XML, the process of tagging must be automated. Currently automatic 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 automatic tagging, we present a novel hybrid system that produces tagged document collections by using a neural network algorithm, the Self-Organizing Map (SOM) [1], [2], and a machine learning algorithm, the inductive learning algorithm C5.0 [3], [4].

2 System overview

The hybrid architecture of our system combines the techniques of the SOM and C5.0 algorithms to produce XML tagged documents. The system architecture is outlined in Figure 1. The first phase of the system deals with the formation of a map of tagged documents using the SOM algorithm. Once a map has been formed, then, in the second phase, an unmarked document is marked up. The system automatically extracts information from the SOM neighbours of the unmarked document in the form of rules by using the inductive learning algorithm C5.0. These rules, together with text segmentation heuristics (also derived from the set of tagged documents) and the rules of the Document Type Definition (DTD) for the document, are used to mark the document into XML.

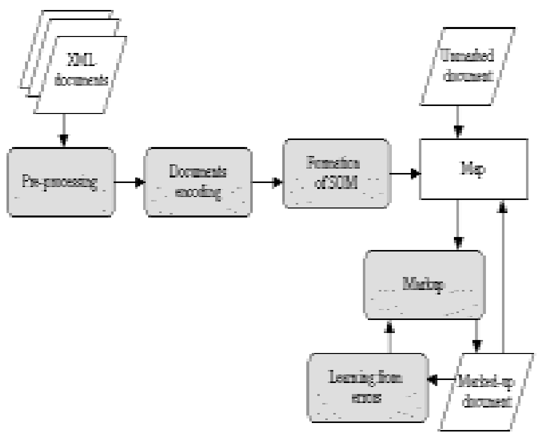

Figure 1. Overview of system architecture.

3 Self-Organising Maps

The SOM algorithm [1], [2] provides a non-linear, ordered, smooth mapping of high-dimensional input data onto a low-dimensional array. This process can be described as follows. Each input variable is represented as a real vector *x(t)* where *t* is the index or discrete time coordinate. With each node in a SOM array, a parametric model vector m_i is associated, the components of which represent the weights. Using a limited number of model vectors all features can be represented on the map with high accuracy. The map algorithm performs a recursive regression process in which only a subset of models is processed at every step. This process produces a smoothing effect on the model vectors in a certain neighbourhood and the process of continued learning results in global ordering of map.

Documents are clustered on the self-organizing map using the WEBSOM algorithm [5], which consists of three stages. In the pre-processing stage, all non-textual information is removed from the text, and all words are stemmed to their basic forms. The most rarely occurring words are also removed. In the document encoding stage, the documents are encoded by forming statistical models. Each document is represented as a real vector in which each component represents the frequency of occurrence of a particular word in the document. Therefore, each document vector can be viewed as a weighted word histogram. In the map formation stage, the set of statistical models or encoded documents is given as input to the SOM algorithm to form a document map. We use SOM_PAK [6] for this purpose. An example of a map created by the system is given in Figure 2.

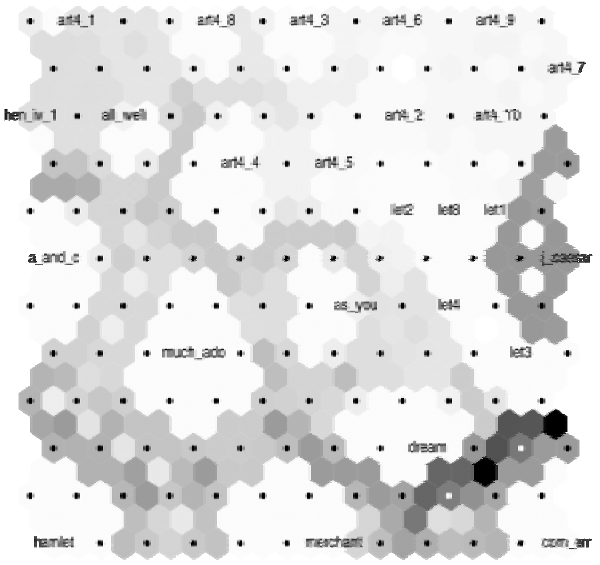

Figure 2. A SOM of different document collections. Shades of grey are used to show document density: lighter shades indicate high density.

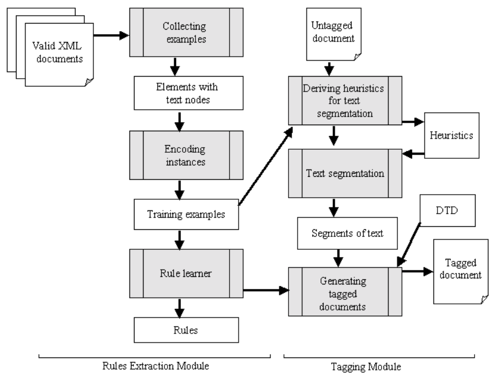

Figure 3. The auto-tagging process.

4 The auto-tagging process

The auto-tagging process (the second phase of the system) has two main modules, a rule extraction module and a tagging module. The process is shown in Figure 3. The rule extraction module learns rules from a collection of tagged documents using an inductive learning approach [7]. Training examples are collected from a set of valid XML documents. These documents are from a specific domain and their markup is valid and complies 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 elements nested in the root. Only elements containing 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 text enclosed between the start and end tags of all occurrences of each element is encoded using the fixed-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 use 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, it can deal with large number of features and it is incremental. C5.0 is best suited for our system because 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 tag the text documents from the same domain.

The second module creates a tagged version of an untagged text document. The untagged document should be from the same domain as the documents used for learning the rules. For the auto-tagging of text document, it 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 C5.0 algorithm and the text segmentation heuristics, the hierarchical structure of the document is obtained and a tagged version of a 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 and do not recognize the content; therefore a human expert is required to evaluate the accuracy of the auto-tagging process.

5 Experiments

For our experiments, we have used collections of documents from a number of different domains. These include letters from the MacGreevy Archive [8], [9], a database of employee records, Shakespearean plays [10], poems from the Early American Digital Archives [11] and scientific journal articles [12].

```
5 Cotham Park
Bristol 6

September 1955

Dear Tom,

I fear my letter to you from Paris was not a
full answer to your letters abut Eupalinos and
I did not enclose Wallace Stevens' letter to
you. Here it is. I am terribly sorry – and all
the more deeply because I just – at last – had
got at this whole complex (which I must
explain to you
more fully [?now]) and was reading for the
first time some of Wallace Stevens own poems
(some
of which are beautiful – though I have not
found the two dedicated to you) when the news
came
...
```

Figure 4. An unmarked letter from the MacGreevy Archive.

Figure 4 contains the unmarked text of a letter from the MacGreevy Archive. The marked-up version of the letter is in Figure 5. The markup produced by our system is mostly correct; the only incorrectly tagged text is that underlined in Figure 5, between the `<INSIDEADDRESS>` and `</INSIDEADDRESS>` tags. The DTD used to mark up the letters in the MacGreevy Archive is given in Figure 6.

Another example of the markup produced by our system, an excerpt from *A Midsummer Night's Dream*, is shown in Figure 7. Again, the markup in this example is largely correct; the only error is the underlined text, between the `<STAGEDIR>` and `</STAGEDIR>` tags.

As a final example, Figure 8 contains part of a scientific article marked up by our system. Again, the incorrectly tagged text, between the `<title>` and `</title>` tags and between the `<orgName>` and `</orgName>` tags, is underlined.

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "LETTER.dtd" >
<LETTER>
<INSIDEADDRESS>
5 Cotham Park
Bristol 6<LINEBREAK/>
</INSIDEADDRESS>
<DATE>September 1955</DATE>
<SALUTATION>Dear Tom,</SALUTATION>
<BODY>
<PARA>I fear my letter to you from Paris was
not a full answer to your letters abut
Eupalinos and
I did not enclose Wallace Stevens' letter to
you. Here it is. I am terribly sorry and all
the more
deeply because I just – at last – had got at
this whole complex (which I must explain to
you
more fully [?now]) and was reading for the
first time some of Wallace Stevens own poems
(some
of which are beautiful – though I have not
found the two dedicated to you) when the news
came
of his death – if you have not had it yet it
will be a great
...
```

Figure 5. The marked-up version of the letter in Figure 3.

```
<!ELEMENT LETTER (INSIDEADDRESS,
      DATE, SALUTATION, BODY,
      CLOSING, SIGNATURE?)>
<!ELEMENT INSIDEADDRESS (#PCDATA |
            LINEBREAK)*>
<!ELEMENT LINEBREAK EMPTY>
<!ELEMENT DATE (#PCDATA)>
<!ATTLIST DATE ALIGN (LEFT | RIGHT)
            "RIGHT">
<!ELEMENT SALUTATION (#PCDATA)>
<!ELEMENT BODY (PARA+)>
<!ELEMENT PARA (#PCDATA)>
<!ATTLIST PARA ALIGN(LEFT |RIGHT|
            JUSTIFY) "LEFT">
<!ELEMENT CLOSING (#PCDATA)>
<!ELEMENT SIGNATURE (#PCDATA)>
```

Figure 6. XML DTD for letters in the MacGreevy Archive.

For the scientific journal articles we have used additional heuristics specifically devised for this domain. However, we expect that these heuristics can be used for articles from most journals. The tagged articles used as training documents for our experiments have been downloaded from the World Wide Web along with the DTD (`article.dtd`) devised for these articles. The XML DTD used for these tagged articles is complicated and requires the presence of another DTD (`biblist.dtd`) devised for references and bibliographies.

6 System evaluation

We use three performance measures to evaluate the performance of our system:

- The percentage of elements correctly tagged by the system (P_c);
- The percentage of elements incorrectly tagged by the system (P_i);
- The percentage of elements not tagged by the system (P_m).

We have evaluated the performance of our system on the different domains. A summary of the results is presented in Table 1.

Domain	**P_c**
Letters from the MacGreevy Archive	96%
Shakespearean plays	92%
Scientific journal articles	97%

Table 1. Summary of system performance on different domains.

7 Conclusions

We have described a novel hybrid architecture for the organization of documents and their markup into XML. Our system uses self-organizing capabilities to organize and explore the XML documents. Furthermore, it provides a generic tool to automatically markup the documents by extracting knowledge from previously marked-up examples. The ability to learn from markup errors promises to make it even more effective as a markup tool for use in producing XML-tagged document collections.

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

```
…
<SCENE>
  <TITLE>SCENE I. Athens. The palace of THESEUS.</TITLE>
  <STAGEDIR>Enter THESEUS, HIPPOLYTA, PHILOSTRATE, and Attendants</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>
…
```

Figure 7. Part of *A Midsummer Night's Dream* marked up by our system.

```
<?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 periodicity only lasted up to a maximum thickness
of 2.5 ML, followed by a transition to the unreconstructed surface. Samples grown under N-
rich, Ga-rich and stoichiometric conditions were characterised by high-resolution scanning
electron microscopy and atomic force microscopy. We found that the smoothest surfaces were
provided by the N/Ga-ratio giving the thickest layer at the (3x3)=&gt;(1x1) transition. The
defect formation at the GaN/GaAs interface also depended on the N/Ga-flux ratio.</p>
  </abstract>
  </front>
  <body>
    <section>
      <heading>Introduction</heading>
      <p>Gallium nitride is one of the most promising materials for optical applications in
the blue range of the visible spectra due to its direct energy band gap of 3.39 eV at
…
```

Figure 8. Part of a scientific journal article automatically tagged by our system.

References:
[1] T Kohonen, “Self-Organized Formation of Topologically Correct Feature Maps”, *Biological Cybernetics*, Volume 43, 1982, pp 59-69.
[2] T Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, 2001.
[3] JR Quinlan, *C4.5: Programs For Machine Learning,* Morgan Kaufmann Publishers, San Mateo, California, 1993.
[4] JR Quinlan, *Data Mining Tools See5 and C5.0*, http://www.rulequest.com/see5-info.html, 2000.
[5] T Kohonen, “Self-Organization of Very Large Document Collections: State of the Art,” In *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, Volume 1, Springer, London, 1998, pp 65-74.
[6] T Kohonen, J Hynninen, J Kangas, J Laaksonen, *SOM_PAK: The Self-Organizing Map Program Package*, Version 3.1, Helsinki University of Technology, Laboratory of Computer and Information Science, 1995.
[7] T Mitchell, *Machine Learning*, McGraw-Hill, 1997.
[8] S Schreibman, *The MacGreevy Archive*, http://www.ucd.ie/~cosei/archive.htm, 1998.
[9] S Schreibman, *The MacGreevy Archive*, http://www.iath.virginia.edu/macgreevy, 2002.
[10] J Bosak, *Shakespeare 2.00*, http://metalab.unc.edu/bosak/xml/eg/shaks200.zip, 1998.
[11] S Schreibman, *Early American Digital Archives*, Maryland Institute of Technology, http://www.mith.umd.edu, 2003.
[12] Openly Informatics, Inc, http://www.openly.com/efirst, 1999-2000.

† Current address: Department of Computer Science, National University of Ireland, Maynooth, Maynooth, County Kildare, Ireland.