Automating XML Markup using Machine Learning Techniques

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

[1]Department of Computer Science,
University College Dublin,
Belfield, Dublin 4, Ireland

[2]Department of Computer Science,
National University of Ireland Maynooth,
Maynooth, Co. Kildare, Ireland

ABSTRACT

In this paper we present a novel system for automatically marking up text documents into XML. The system uses the techniques of the Self-Organising Map (SOM) algorithm in conjunction with an inductive learning algorithm, C5.0. The SOM algorithm clusters the XML marked-up documents on a two-dimensional map such that documents having similar content are placed close to each other. The C5.0 algorithm learns and applies markup rules derived from the nearest SOM neighbours of an unmarked document. The system is designed to be adaptive so that it learns from errors in order to improve the markup of resulting document. Experiments shows that our system provides high accuracy and demonstrate that our approach is practical and feasible.

Keywords: Automatic Markup, XML, Machine Learning, C5.0, Self-Organizing Map.

1. INTRODUCTION

With the extraordinary growth of the World Wide Web, large amounts of information are now available in electronic form. With such an explosion of online information, the ability to manage vast and complex repositories of data and to search for relevant information is becoming increasingly more difficult. Existing information retrieval systems are useful for locating relevant documents but are not effective for answering content-based queries. Therefore it is necessary to develop intelligent data management and retrieval techniques. Currently HTML is the most widely used markup language on the Web but it lacks extensibility, structure and validation. With the fixed tag set of HTML, the logical structure of documents cannot be represented and the semantic content of HTML pages cannot be extracted for different applications. Due to the limitations of HTML, XML has emerged as a new standard for intelligent document management and electronic publishing. XML is simpler than SGML but more powerful than HTML. It is flexible and portable and can represent the hierarchical structure of documents. By using XML, users can define their own tags appropriate for an application domain. XML separates content from style issue, making it useable for a variety of formats and applications. Content-based searches can be performed by exploiting the XML structure of documents, providing an opportunity for the development of more powerful intelligent information retrieval systems. Although XML is very popular and is receiving widespread adoption, we still do not have large collections of XML data. Manual XML markup is tedious, requiring a lot of time and effort; hence it is impractical to produce tagged collections without automation. Some commercial text handling systems (e.g., Panorama, XMetal) provide support for marking up documents in SGML and XML, but these require a considerable amount of manual effort. Automatic XML markup is still a significant challenge. However, most systems that have been developed are limited to specific domains and require human intervention. In addressing the need of automatic XML markup, we present a system for marking up text documents using a combination of Self-Organising Map algorithm [1, 2] and an inductive learning algorithm, C5.0 [3, 4].

2. SYSTEM ARCHITECTURE

Our system is designed as a hybrid system and combines the techniques of the SOM and C5.0 algorithms for the adaptive automatic markup of text documents into XML. The hybrid system is shown in Figure 1.

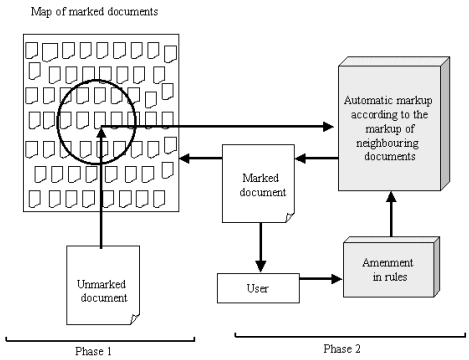

Figure 1: System Architecture: Phase 1 deals with the formation of the Self-Organising Map using the WEBSOM method. Phase 2 deals with the markup of an incoming unmarked document using the inductive learning algorithm, C5.0.

The first phase of the system deals with the formation of a map of a collection of marked-up documents, using the WEBSOM method [5]. WEBSOM is an application of SOM to document classification and retrieval. WEBSOM arranges a large collection of documents on a two-dimensional grid using the SOM algorithm. SOM is a neural network-based unsupervised learning algorithm, which maps higher-dimensional statistical data onto a lower-dimensional grid or *map* such that similar documents appear close to each other on the map.

The second phase of the system deals with the automatic markup of text documents and is implemented as an independent XML markup system, which is described in section 3. Eventually these two phases will be combined to form an integrated hybrid system, which is described below.

Once a map of marked-up documents has been formed, a new incoming document will be mapped to the cluster of documents most similar to it on the Self-Organising Map. The system then captures the markup information from the elements of neighbouring marked-up documents, and learns classifiers by using an inductive learning algorithm. The incoming document will then be automatically marked-up according to the learned classifiers. If the document is not related at all to the existing documents on the map, it will be discarded. The system has an adaptive behaviour and it learns from errors to improve the markup.

3. AUTOMATING THE PROCESS OF XML MARKUP

The automatic XML markup process has two main modules; a rule extraction module and a markup module, as shown in Figure 2.

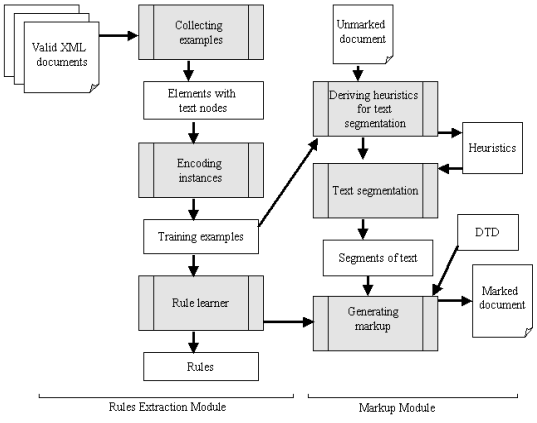

Figure 2: Automating the process of XML markup

The rule extraction module uses an *inductive learning* [6] approach to learn rules from a collection of marked-up documents. In this module, *training examples* are gathered from a collection of valid XML marked up documents, which are encoded using a fixed-width feature vector. The inductive learning algorithm processes all these encoded examples to learn *classifiers*, which are used in the task of marking up an unmarked document.

Collecting examples

The 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 represents a tree-like structure having a root *element* and other nested *elements*. The leaf elements containing text are considered as *markup elements* for our system. For example, a marked-up XML letter from the MacGreevy archive [7, 8] is shown in Figure 3. This letter is marked up according to the DTD shown in Figure 4.

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "LETTER.dtd">
<LETTER>
  <DATE>October 15th [postmarked
1935]</DATE>

<INSIDEADDRESS> 15, CHEYNE
GARDENS,<LINEBREAK/>
CHELSEA, <LINEBREAK/>
S.W.3. </INSIDEADDRESS>

  <SALUTATION>My dear Tom,</SALUTATION>

  <BODY>
     <PARA>I am very glad you are less
anxious.  </PARA>
     <PARA>I have been extremely busy since
I got back & worried!  Now the income tax
people have been at me just as Teddy was
temporarily silenced. </PARA>  </BODY>

  <CLOSING>Ever affectionately</CLOSING>

  <SIGNATURE>Hester</SIGNATURE>
</LETTER>
```

Figure 3: A letter from the MacGreevy Archive [7, 8]

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

Figure 4: DTD for marking up letters from the MacGreevy Archive [7, 8]

For the collection of letters from the MacGreevy Archive [7, 8], `DATE`, `INSIDEADDRESS`, `SALUTATION`, `PARA`, `CLOSING`, `SIGNATURE` are considered as markup elements and classifiers are learned for these elements while other elements such as `LETTER` and `BODY`, which contain nested

elements as children, are marked-up by using the appropriate DTD only. Each *training instance* corresponds to a leaf element containing text from the collection of marked-up documents. One training example from the collection of letters is shown below:

```
<DATE>11th October, 1940</DATE>
```

The text enclosed between the start and end tag of all occurrences of each element is encoded using the fixed-width feature vector. These encoded instances are used subsequently for learning the rules. The encoding of instances is described in the next section.

Encoding Instances

In our system rule extraction for the automatic XML markup is considered as an inductive learning task. Thirty-one features, such as word count, character count, etc, are used to encode the training instances. The set of *encoded instances*, which is pre-classified by the name of the element, is used by the system to learn classifiers for those elements with that tag names. These classifiers are later used for marking up the text of an unmarked document.

Rule Learner

The inductive learning algorithm processes the encoded instances to develop classifiers to convert the text document into an XML marked-up document. For this purpose we have selected the C5.0 learning algorithm. 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. Our system deals with documents from different domains, so some of the features are not relevant to the documents of all domains. For these reasons C5.0 is best suited for our system. Sets of rules are generated in a given domain from a collection of XML marked-up documents and are used to markup the text documents from the same domain. The numbers of rules learned by the C5.0 classifier for each document set is different. For example, C5.0 learned a set of seven rules for the letters from the MacGreevy Archive [7, 8]. A paraphrase of a rule learned from the set of letters is given below:

If the text segment contains the name of the month and up to six white space characters then it is specified as the text node of the `DATE` *element.*

Markup

The second module deals with the conversion of a text document into an XML marked-up document. The unmarked document should be from the same domain as the documents used for learning the rules. To accomplish the markup, the unmarked 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 and by applying the rules extracted by C5.0 on the stored text segments, the hierarchical structure of the document is encoded and the document is marked-up in XML.

The markup produced by the system can be validated against the DTD by using any XML parser. However, in order to evaluate the accuracy of the markup, a human expert should examine it because XML processors are not content-sensitive and only validate the syntax of the marked-up document.

4. EXPERIMENTS

We have used document sets from a number of different domains. An example of marking up a document taken from a set of employee records is shown in Figure 5.

a)

```
Randall
Atkinson
atkinson@itd.nr
l.navy.mil

Mark Brader
msb@sq.com

Alessio Dragoni
drago@ats.it

Patrick
Hoepfner
hoepfner@heasfs
.gsfc.nasa.gov

Dan Hoey
hoey@aic.nrl.na
vy.mil

Kjetil Torgrim
Homme
kjetilho@ifi.uio
.no

 Jonathan I.
Kamens
jik@security.ov
.com

Mark Kantrowitz
mkant@cs.cmu.ed
u
```

b)

```
<?xml version="1.0"?>
<!DOCTYPE department
SYSTEM department.dtd">
<department>
  <employee>
     <name>Randall
Atkinson</name>
<email>atkinson@itd.nrl.n
avy.mil</email>
  </employee>

  <employee>
    <name>Mark
Brader</name>

<email>msb@sq.com</email>
  </employee>

<employee>
    <name>Alessio
Dragoni</name>
<email>drago@ats.it</emai
l>
  </employee>

<employee>
 <name>Patrick
Hoepfner</name>
<email>hoepfner@heasfs.gs
fc.nasa.gov
</email>
  </employee>

<employee>
 <name>Dan Hoey</name>
<email>hoey@aic.nrl.navy.
mil</email>
  </employee>

<employee>
   <name>Kjetil Torgrim
Homme</name>
<email>kjetilho@ifi.uio.n
o</email>
  </employee>

  <employee>
     <name>Jonathan I.
Kamens</name>
<email>jik@security.ov.co
m</email>
  </employee>

  <employee>
    <name>Mark
Kantrowitz</name>
<email>mkant@cs.cmu.edu</
email></employee>
</department>
```

Figure 5: XML markup of a document of employee records a) original text document b) marked-up document

The marked up documents used for learning rules from this domain conform to the DTD department.dtd [9]. This is an example of a document which uses a very simple DTD. The system works very well with simple DTDs. Another set of documents used was the "Letters" from the MacGreevy Archive [7, 8]. The text version of a letter is shown in Figure 6 and the marked-up version of the same letter is shown in Figure 7.

```
19, WESTGATE TERRACE,
REDCLFFESQURE.
S.W 10.

Friday 7th July 1936

Dear Mr McGreevy

I heard from the lawyer today, and I am glad
to say that he will be
able to make some payments in advance to
Miss H. Dowden -

  Yrs sincerely

C.E. Harrison
```

Figure 6: A letter taken from the MacGreevy Archive [7, 8]

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "letter.dtd">
<LETTER>
  <DATE> Friday 7th July 1936</DATE>

  <INSIDEADDRESS> 19, WESTGATE
   TERRACE,<LINEBREAK/>
REDCLFFESQURE.<LINEBREAK/>
S.W.10.<LINEBREAK/>
</INSIDEADDRESS>

  <SALUTATION> Dear Mr McGreevy
</SALUTATION>

  <BODY>
    <PARA> I heard from the lawyer today,
and I am glad to say that he will be able to
make some payments in advance to Miss H.
Dowden -</PARA>
  </BODY>

<CLOSING> Yrs sincerely</CLOSING>

  <SIGNATURE> C.E. Harrison
   </SIGNATURE>
</LETTER>
```

Figure 7: Marked-up version of the letter shown in Figure 6

Another domain that we have worked with is Shakespearean plays [10]. An example of a part of a scene from "Cymbeline" marked-up by our system is shown in Figure 8. The underlined text shown in Figure 8 is not marked up by our system.

```
…
<SCENE>
      <TITLE>SCENE III.  A room in Cymbeline's
palace. </TITLE>

<SPEECH>
         <STAGEDIR>Enter IMOGEN and
PISANIO</STAGEDIR>
        <SPEAKER>IMOGEN</SPEAKER>
        <LINE>
          I would thou grew'st unto the shores
o' the haven,</LINE>
        <LINE>And question'dst every sail: if
he should write</LINE>
        <LINE>And not have it, 'twere a paper
lost,</LINE>
        <LINE>As offer'd mercy is. What was
the last</LINE>
        <LINE>That he spake to thee?</LINE>
      </SPEECH>

<SPEECH>
        <SPEAKER>PISANIO</SPEAKER>
        <LINE>
          It was his queen, his queen!</LINE>
        </SPEECH>
      <SPEECH>
        <SPEAKER>IMOGEN</SPEAKER>
        <LINE>
          Then waved his handkerchief?</LINE>
        </SPEECH>
      <SPEECH>
        <SPEAKER>PISANIO</SPEAKER>
        <LINE>
          And kiss'd it, madam.</LINE>
        </SPEECH>

<SPEECH>
        <SPEAKER>IMOGEN</SPEAKER>
        <LINE>
          Senseless Linen! happier therein
than I!</LINE>
        <LINE>And that was all?</LINE>
        </SPEECH>
      <SPEECH>
        <SPEAKER>PISANIO</SPEAKER>
        <LINE>
          No, madam; for so long</LINE>
        <LINE>As he could make me with this
eye or ear</LINE>

        <LINE>Distinguish him from others, he
did keep</LINE>
        <LINE>The deck, with glove, or hat, or
handkerchief,</LINE>
        <LINE>Still waving, as the fits and
stirs of 's mind</LINE>
        <LINE>Could best express how slow his
soul sail'd on, </LINE>
        <LINE>How swift his ship.</LINE>
      </SPEECH>

<SPEECH>
        <SPEAKER>IMOGEN</SPEAKER>
        <LINE>
       …
```

Figure 8: A part of scene taken from "Cymbeline"

The Shakespearean plays are large documents and the DTD used for plays is complicated, but the system still performs well

with the plays, although the performance is not as good as it is for documents with simple DTDs.

Another set of documents with which we have worked is taken from "Early American Digital Archives" [11]. Part of a poem encoded by our system is shown in Figure 9. As before, the text not marked-up by our system is underlined.

```
…
<text>
   <body>
      <div0>
         <head>
           <title>Upon My Dear and Loving
Husband his Going into England</title>
</head>
         <lg> <l>O thou Most High who rulest
all</l>
             <l>And hear'st the prayers of
thine,</l>
            <l>O hearken, Lord, unto my
suit</l>
            <l>And my petition sign.</l> </lg>
            <lg><l>Into Thy everlasting arms
Of mercy</l>
            <l>I commend Thy servant,
Lord.</l>
         <l>Keep and preserve My husband,</l>
         <l>my dear friend.</l></lg>
         <lg> <l>At Thy command, O Lord, he
went,</l>
       <l>Nor nought could keep him back.</l>
     <l>Then let Thy promise joy his
heart,</l>
       <l>O help and be not slack.</l> </lg>
…
```

Figure 9: Part of a poem marked-up by our system

5. EVALUATION

We have used three performance measures in evaluating the automatic markup process.

- The percentage of marked-up elements correctly determined by the system
- The percentage of marked-up elements incorrectly determined by the system
- The percentage of marked-up elements not determined by the system (i.e. text nodes for these marked-up elements are not present in the marked-up document).

We use the term "accuracy" here to refer to the first of these measures, i.e. the percentage of marked-up elements correctly determined by the system.

In an earlier version of the system we worked with a set of well-formed documents, namely letters from the MacGreevy Archive [7, 8]. We have evaluated our system by using 211 elements of twenty letters from the MacGreevy Archive and achieved 94% accuracy. In the current version of the system we are working with validly marked-up documents. We have evaluated valid XML marked-up letters produced by our system and achieved 96% accuracy. The valid XML letters are also taken from the MacGreevy Archive and comply with the rules of the DTD shown in Figure 4. For the Shakespearean plays our system achieves 92% accuracy, which we consider to be a good performance.

6. CONCLUSIONS

We have described a system with novel hybrid architecture for marking up text documents into XML. The system collects the markup information from neighbouring documents on the Self-Organising Map. The information is extracted in the form of rules by using the C5.0 learning algorithm. The rules are then applied to unmarked text documents to produce markup. The good results from our experiments demonstrate that our approach is practical and feasible.

7. ACKNOWLEDGEMENTS

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

8. REFERENCES

[1] T. Kohonen, "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, 1997.

[2] T. Kohonen, **Self-Organizing Maps,** Springer Series in Information Science, 2001.

[3] J. R. Quinlan, **C4.5: Programs for Machine Learning**. Morgan Kauffmann, Los Altos, CA, 1993.

[4] J. R. Quinlan, **Data Mining Tools See5 and C5.0,** URL:[http://www.rulequest.com/see5-info.html], 2002.

[5] T. Honkela, S. Kaski, K. Lagus, & T. Kohonen, "Newsgroup Exploration with WEBSOM method and browsing", **Technical Report A32, Helsinki University of Technology,** Laboratory of Computer and Information Science, Espoo: Finland, 1996.

[6] T. M. Mitchell, **Machine Learning,** McGraw-Hill, NewYork, NY, 1997.

[7] S. Schreibman, **The MacGreevy Archive,** URL: [http://www.ucd.ie/~cosei/archive.htm], 1998.

[8] S. Schreibman, **The MacGreevy Archive,** URL: [http://jafferson.village.virginia.edu/macgreevy], 2000.

[9] H. Maruyana, K. Tamura, N. Uramoto, **XML and Java: Developing web applications,** Addison-Wesley, 1999.

[10] J. Bosak, **The Plays of Shakespeare in XML**, URL: [http://www.oasis-open.org/cover/bosakShakespeare200.html], 1999.

[11] S. Schreibman, **Early American Digital Archives,** Hosted by Maryland Institute of Technology. URL: [http://www.mith.umd.edu], 2003.