Transformation-Based Learning for Automatic Translation from HTML to XML

James R. Curran
Basser Department of Computer Science
University of Sydney
N.S.W. 2006, Australia
jcurran@cs.usyd.edu.au

Raymond K. Wong
Basser Department of Computer Science
University of Sydney
N.S.W. 2006, Australia
wong@cs.usyd.edu.au

Abstract

Format tags implicitly represent content information in the same ambiguous, context dependent manner that words represent semantics in natural language. Translation from format to content markup shares many characteristics with tagging and parsing tasks in computational linguistics. The transformation-based learning (TEL) paradigm has recently been applied to numerous computational linguistics tasks with considerable success. We present a transformation-based translator which automatically learns to translate semistructured HTML documents formatted with a particular style to XML using a small set of training examples.

Keywords HTML, XML, markup, document processing, machine translation, machine learning

1 Introduction

XML [1] is the focus of increasing interest in markup representing *content* rather than *form*. Content tags describe the structural semantics of a document explicitly while form tags represent semantics implicitly as a function of document formatting. For instance, the XML tag <TITLE> describes a title, whereas the HTML tag <B> describes a bold font which may represent a title or one of many other document structures.

The XML revolution is resulting in huge quantities of legacy format tagged documentation in HTML, postscript, and word processor files which will need to be converted. There is also a need to convert between XML languages and recover the XML from XML documents formatted using XSL. Automatic methods for translation from format to content tagged documents are therefore of considerable research and commercial value.

The structural semantics represented by formatting are ambiguous, and context dependent, like the representation of semantics by natural language. For instance, Part of Speech (POS) tagging describes the role that each word plays in a sentence, such as noun (NN) or adjective (JJ)[1]. Each tag is co-dependent on the surrounding words and their tags. The TBL paradigm, proposed in [4], has been successfully applied to POS tagging [2, 5, 6]. Other TBL applications include text segmentation and classification [11, 9] and prepositional phrase attachment [7] which all involve learning annotations with extent which is necessary for HTML to XML translation. This work extends our TBL linear format to content tag translation, [8], where format and content tags are flattened not nested.

2 Transformation-Based Learning

Transformations are rules that are applied in a given context to change a tag from an erroneous value to its correct value. TBL automatically acquires a sequence of transformations to incrementally patch tagging errors. Transformations are applied to a simple initial guess at the correct tag. The POS tagger [2] initial guess was the most common tag for each word which was extracted from the training corpus. For TBL syntactic parsing [3], a right branching bracket structure, which is most common in English, was used as the initial state. The initial state does not need to use domain dependent knowledge and may represent no useful information such as the NULL tag.

A transformation modifies a tag when the context (such as neighbouring tags or words) of the tag matches the context described by the transformation. Every transformation is applied in sequence to every tag site in the document. The first ten POS tagging transformations from [2] are shown in Figure 1.

The transformation learning algorithm is illustrated in Figure 2. The training corpus is first stripped of the annotations that the tagger will learn. This stripped corpus is then annotated with the naive initial tagging which has been derived from statistical analysis of the training corpus or limited domain knowledge. The TBL algorithm

[1] Appendix A describes each POS tag

Proceedings of the Fourth Australasian Document Computing Symposium, Coffs Harbour, Australia, December 3, 1999.

#	tag	to	when
1	NN	VB	prev. tag is TO
2	VBP	VB	one of the prev. 3 tags is MD
3	NN	VB	one of the prev. 2 tags is HD
4	VB	NN	one of the prev. 2 tags is DT
5	VBD	VBN	one of the prev. 3 tags is VBZ
6	VBN	VBD	prev. tag is PRP
7	VBN	VBD	prev. tag is NNP
8	VBD	VBN	prev. tag is VBD
9	VBP	VB	prev. tag is TO
10	POS	VBZ	prev. tag is PRP

Figure 1: Transformations for POS tagging [2]

attempts to duplicate the training corpus from the initial tag state by iteratively learning transformations which patch errors in the current tag state. In each iteration, shown highlighted in Figure 2, the best scoring transformation in the current state is learned. From the current state, training corpus and transformation templates, a set of possible transformations is proposed. The error driven model uses each erroneous tag context to propose transformations by template instantiation. Each transformation is then evaluated using the training corpus and evaluation function. In the POS tagger [2], the evaluation function was the net accuracy improvement as a result of applying the transformation. The highest scoring transformation is appended to the learnt sequence and applied to the current state. Iterations continue until the desired tagging accuracy has been reached or none of the proposed transformations have a positive score.

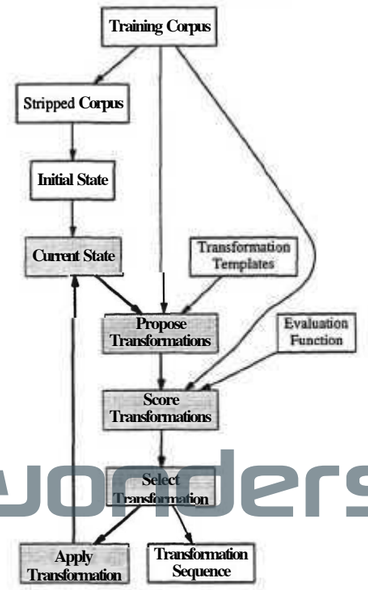

Figure 2: The transformation learning process

The transformation search space is limited by transformation templates. Templates describe valid transformations and must describe properties that will reliably indicate when a transformation is applicable. If a context relationship is not described by a transformation template then it cannot be represented by the system. Proposed transformations are formed using templates instantiated by filling their slots with the context of the erroneous tags. The POS tagging transformation templates from [5], shown in Figure 3, are primarily simple bigram and trigram relationships between neighbouring words and tags.

Change tag A to tag B when:

1. the *preceding (following)* word is tagged Z
2. the word *two before (after)* is tagged Z
3. *one* of the *two preceding (following)* words is tagged Z
4. *one* of the *three preceding (following)* words is tagged Z
5. the *preceding* word is tagged Z and the *following)* word is tagged Y
6. the *preceding (following)* word is tagged Z and the word *two before (after)* is tagged Y

Figure 3: Templates for POS tagging [5]

3 HTML to XML

A transformation-based learning system is defined by the:

- *initial state* that is applied to the document to be tagged
- *evaluation function* that is used during training to evaluate the quality of the proposed transformations
- ***set** of transformation templates* that define the context relationships that learned transformations can represent

We define these system parameters for HTML to XML translation in the following sections but first discuss preprocessing of the HTML and XML files.

A typical problem is the translation of a list of published papers on an academic's web page into XML (or bibtex) so that it can be loaded into an object database of research papers. Figure 4 and Figure 5 are fragments of HTML and XML that may be part of the training and target data for this problem. We will use examples from these fragments as motivation for the design of our system.


```
1 <H1><A NAME="SECTIDN000400000000000000000"X/A>
2 <IMG SRC="OLD/blueball.gif" HEIGHT=14 WIDTH=14 ALIGN=BOTTOM>
3 <FONT COLOR="#00FFFF">Publications</FONT></H1>
4 <FONT COLOR="#0000FF"><FONT SIZE=-2></FONT></FONT>
5 <UL>
6 <LI><B>R.K. Wong</B>, H.L. Chau, and F.H. Lochovsky.
7 <FONT COLOR="#FFFF00">Dynamic Knowledge Representation in DOOR.</FONT>
8 In IEEE KDEX Workshop,Newport Beach, CA, November
9 1997.<A HREF="Papers/design.ps">(The full postscript (journal) version)</A> </LI>
10 <LI><B>R.K. Wong</B>, H.L. Chau, and F.H. Lochovsky.
11 <FONT COLOR="#FFFF00">A Data Model and Semantics of Objects with Dynamic Roles.</FONT>
12 <I>1997 IEEE International Conference on Data Engineering</I>, pp.402-411, UK,
13 April 1997.<A HREF="Papers/icde97.ps">(The postscript file)</A> </LI>
14 </UL>
```

Figure 4: A fragment of HTML from the Publications translation task

```
1  <Publications>
2    <paper>
3      <author> R.K. Wong </author>
4      <author> H.L. Chau </author>
5      <author> F.H. Lochovsky </author>
6      <title> Dynamic Knowledge Representation in DOOR </title>
7      <workshop> IEEE KDEX Workshop </workshop>
8      <location> Newport Beach, CA </location>
9       <date> November 1997 </date>
10     <softcopy HREF="Papers/design.ps" />
11   </paper>
12   <paper>
13     <author> R.K. Wong </author>
14     <author> H.L. Chau </author>
15     <author> F.H. Lochovsky </author>
16     <title> A Data Model and Semantics of Objects with Dynamic Roles </title>
17     <conference> IEEE International Conference on Data Engineering </conference>
18     <location> UK </location>
19     <pages> 402-411 </pages>
20     <date> April 1997 </date>
21     <softcopy HREF="Papers/icde97.ps" />
22   </paper>
23 </Publications>
```

Figure 5: XML fragment corresponding to Figure 4

```
1  <H1>
2  <IMG HEIGHT=14 WIDTH=14 ALIGN=B0TT0MXSRC>0LD/blueball.gif</SRCX/IMG>
3  <FONT COLOR="#00FFFF">Publications</FONT></H1>
4
5  <UL>
6  <LIXB>[R.K. Wong]</B>, [H.L. Chau], and [F.H. Lochovsky].
7  <FONT COLOR="#FFFF00">[Dynamic Knowledge Representation in DOOR].</F0NT>
8  In [IEEE KDEX Workshop],[Newport Beach, CA], [November
9  1997].<A><HREF>[Papers/design.ps]</HREF>(The full postscript (journal) version)</A> </LI>
10 <LIXB>[R.K. Wong]</B>, [H.L. Chau], and [F.H. Lochovsky].
11 <FONT COLOR="#FFFF00">[A Data Model and Semantics of Objects with Dynamic Roles].</F0NT>
12 <I>1997 [IEEE International Conference on Data Engineering]</I>, pp.[402-411], [UK],
13 [April 1997].<A><HREF>[Papers/icde97.ps]</HREF>(The postscript file)</A> </LI>
14 </UL>
```

Figure 6: Preprocessed HTML with aligned segments

3.1 Preprocessing

The transformation process begins by removing HTML tags that do not provide any useful information such formatting that does not apply to any text or targets within the text. In Figure 4, both line 4 and <A NAME="SECTION00040000000000000000"></A> can be removed without loss of information. Formatting is removed because transformation templates define contexts of limited length which means that significant HTML tags may be obscured by extraneous tags, which are commonly found in HTML generated by web page authoring software.

Images may be removed, though it is difficult to decide if an image (such as a ball used as a point marker) represents document structure. The dimensions of the image and frequency of use within the file may give some indications as to the purpose of the image. Further, removing images is dangerous because they may be part of the XML that is produced if they are illustrations. Using image information is outside the scope of our current system.

Some attributes such as HREF within A tags need to be extracted. For consistency, we convert these directly to a pseudo-HTML canonical form, with HREF as a child tag within the A markup. Only attributes that reference external entities such as URL's should be converted to the canonical form. Other attributes should remain in the form of PROPERTY=VALUE. Figure 6 shows the attributes SRC and HREF converted. Comparing URLs in the HTML and XML may allow more intelligent processing.

3.2 Alignment and Chunking

After preprocessing, the text within the HTML and XML documents is aligned. Alignment is performed in a single linear scan, since the XML document only contains a subset of the text (not tags) from the HTML document. After alignment, it is obvious that some words, phrases and punctuation have been discarded in the XML. Conjunctions, key words and punctuation in structured, regular text play a significant role in determining the content of a document. The non-aligned segments in Figure 4 are words and, In, 1997, and pp, phrases in round brackets, and commas and periods. These words are often not aligned because the XML tags make them superfluous by defining the content directly. For this reason, we consider non-aligned words with significance to format tags. Keywords may also be found by searching for the XML tags in the HTML text itself. For instance, if the words conference or workshop appear frequently within the tags <conf erence> or workshop then they are probably significant words in determining the XML label.

Aligned text surrounded by XML tags is considered a text chunk in training. These chunks are marked using square brackets in Figure 6. Although this information is not available in target documents we can still utilise this segmentation to learn transformations as long as the transformation context does not refer to chunk boundaries. Using text segmentation tools, which should be fairly accurate with formatting information available, we may be able to include text chunks in the transformation contexts. This is particularly important for applications where formatting is sparse or does not cover information within the text itself that needs to be marked up. Phrase segmentation and identification have been implemented using the TBL paradigm [9, 11] and our system shares some common template types with these systems.

The text chunk boundaries identified using alignment with the XML training data form the skeleton XML tree structure. XML crystalises at the chunk boundaries where tags are added, substituted, removed, and reordered. Although transformations are learnt to manipulate the tags at the boundaries, they do not refer to these boundaries.

4 Initial State Tagging

The tagger will not know the boundaries of the text chunks where the XML tags should be attached, except when an accurate text segmentation tool has been applied first. For this reason, the initial state tagging cannot make a simple guess at the correct tag or its location, so a NULL initial state is used.

Using a good guess initial state has the advantage of reducing the number of transformations that a system is required to learn to reach the desired level of accuracy. Although there is no practical good guess for this system, the larger number of rules will not be much of a problem because the training sets are not large and the efficiency of the finite state transducer (FST) tagger implementation [10] is not dependent on the number of rules, although it does increase the size of the transducer.

5 Evaluation Function

Choice of the evaluation function is crucial to the quality and subtlety of the set of transformations learnt by the system. To acquire subtle transformations, the evaluation function must differentiate between small changes in the current state that indicate it is closer to the training corpus. To learn quality transformations, that can duplicate all structural aspects of the desired result, the evaluation function must be able to take into account all of these structural features.

The evaluation function for syntactic parsing [3] measures the correctness of a bracketing structure as the percentage of constituents (strings between matching brackets) from the current state that do not cross any constituents of the correct bracketing. This is not a problem in our system since we already have text chunks as a result of alignment. However, we still need to consider non-aligned words erroneously appearing within XML markup and the incorrect ordering of multiple markup tags at each chunk boundary.

The total evaluation score is the sum of the error function at every chunk boundary. The error function at each chunk boundary is:

1. for every missing opening or closing markup tag, add 10 to the error
2. for every extra opening or closing markup tag, add 10 to the error
3. for every non-aligned word that is within a markup pair, add 1 to the error
4. for every pair of tags that is in the wrong order, add 1 to the error

Figure 7: HTML to XML transformation evaluation function

Transformations are learned to minimise this evaluation function. The most errors will be patched by transformations that add the most frequent tags in the least ambiguous contexts. The most frequent tags primarily appear in the leaves of the XML tree structure which means that the XML is often constructed from the most nested layers outwards.

We will continue to experiment with the evaluation function in Figure 7 to improve transformation reliability and generality by taking other features into account, such as the tags each transformation refers to. For instance, the <H1> HTML tag will probably provide us with less ambiguous structural information, further up the tree structure than the <I> tag. There may be an advantage of starting from the least nested XML tags because there is less ambiguity and once tags are added to chunk boundaries they can be used in other transformations to further reduce the ambiguity. The training algorithm can be coerced into adding XML tags to the least nested chunks first by penalising transformations according to the depth of the XML they manipulate.

6 Transformation Templates

Transformation templates must describe various relationships between the elements of the current state which reliably indicate whether the transformation should be applied at that site. These elements include beginning and end markers, content and format tags, and words and punctuation within the HTML document. The possibly large number of format tags per content tag, may cause the format tags to obscure the content tags from transformations if considered equal. We have resolved this by expanding the proximity relationship. A content tag is *directly next* to a tag if it touches the tag and is *indirectly next* to a content tag if it is the first content tag within a given number of tags. The transformation templates are given in Figure 8.

insert ctag X when:

1. the (next,prev) (htag,naw) is Y
2. the (direct,indirect) (next,prev) ctag is Y
3. the (next,prev) word is w
4. the (next,prev) two (htag,ctag,naw) are Y and Z
5. the prev (htag,ctag,naw) is Y and the next (htag,ctag,naw) is Z

replace ctag X with (Y,NULL) when:

1. the (next,prev) htag is Z
2. the (direct,indirect) (next,prev) ctag is Z

remove (word,punctuation) w when:

1. the (next,prev) (htag,naw) is Y
2. the (direct,indirect) (next,prev) ctag is Y
3. the (next,prev) two (htag,ctag,naw) are Y and Z
4. the prev (htag,ctag,naw) is Y and the next (htag,ctag,naw) is Z

swap ctags X Y when: the (next,prev) (ctag,htag,naw) is Z.

Figure 8: Transformation templates for HTML to XML conversion. htag = HTML tag, xtag = XML tag, naw = non-aligned word or punctuation.

Most TBL systems have used a non-NULL initial state with transformations that substitute tags rather than add and delete them. Our system must manipulate multiple tags at each chunk boundary, and because of the empty initial state, must be able to change the number of tags. Transformations must be able to insert tags before, after and

between format and content tags, remove tags, puctuation and words and swap the order of content tags.

Our system has many more transformation templates than previous TBL systems. A large number of transformation templates means that the search space is large. However, since the size of the training data is much smaller than previous applications and the maximum number of errors in the current state is relatively small, a larger search space is not a problem.

7 Cleaning Up

Once all of the learned transformations have been applied in sequence, there are numerous XML tags embedded within the text and between the tags of the HTML. Transformations will have already removed the excess non-aligned text. The HTML tags are then removed leaving the XML document with XML tags and correct text chunks. Our current system does not deliberately balance begin and end XML tags. However, it is implicit in error minimisation that matching tags will be learned. The result must then be parsed with a DTD and corrected if necessary.

8 Conclusion

We have presented the first general, automatic translation system from HTML to XML based on Brill's transformation-based learning paradigm. Work in TBL POS tagging and other linguistic annotation tasks has contributed various techniques that can be applied to the transformation of HTML into XML. Initial processing is applied to the document to translate HTML tags into canonical form and to remove formatting tags that do not contribute any structural information.

A system based on the TBL paradigm is defined by the initial state tagging, error function and the set of transformation templates. We base our learning on alignment of the text chunks within the HTML and XML markup. The initial state tagging does not apply any tags. Tags are added to the beginning and end of each text chunk that is aligned between the HTML and XML training data. The error function evaluates the number of correct and incorrect tags and their order at each beginning and end of each text chunk and the number of correct words in each aligned text chunk. The transformation templates include simple templates to add/remove/reorder content tags depending on context of format and content tags, keywords and other words and punctuation within the training data. Words and punctuation that can be used as part of the segmentation process such as conjunctions and commas are automatically identified by the fact that they do not appear in the aligned XML data. Other transformation templates are used to change the size of the text chunks surrounded by each XML tag to remove unnecessary punctuation, conjunctions and extraneous words.

The final step of the process is to extract the HTML tags leaving just the XML and the text chunks in between. The HTML that is extracted from the XML may be used to generate an XML style file. Future work will involve investigating this possibility. The XML is then converted from the canonical form into the desired form using other simple transformations.

Our HTML to XML conversion system is very flexible and can cope with documents with extensive or limited structure. The tagger can be converted to an optimally efficient FST [10], and the learning algorithm requires only a small training set. This algorithm can be applied in a bootstrap manner: Initial training begins on a small manually tagged corpus; The tagger is then used to translate a larger corpus, which is then corrected and used to retrain the tagger. This train-correct cycle continues until the desired accuracy and coverage is achieved. This method greatly reduces the cost of manually tagging training data. Many legacy HTML documents will need to be migrated to XML, ensuring the continuing research and commercial interest in flexible automatic solutions. Used in conjunction with automatic linguistic tools, such as a POS taggers and shallow parsers, format conversion forms the basis of an automatic text summarisation and information retrieval system that generates XML directly. Future research will explore such possibilities.

References

[1] T. Bray, J. Paoli and CM. Sperberg-McQueen (editors). *Extensible Markup Language (XML) 1.0*. Word Wide Web Consortium, 1998. REC-xml-19980210.

[2] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[3] E. Brill. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Meeting of the Association for Computational Linguistics*, Columbus, Oh., 1993.

[4] E. Brill. *A Corpus-Based Approach to Language Learning*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, 1993.

[5] E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Wa., 1994.

[6] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 1995.

[7] E. Brill and P. Resnik. A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan, 1994.

[8] J.R. Curran and R.K. Wong. Transformation based learning in document format processing. In *Working notes of the AAAI 1999 Fall Sypo-sium on Using Layout for the Generation, Understanding or Retrieval of Documents*, 1999.

[9] L. Ramshaw and M. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, 1995.

[10] E. Roche and Y. Schabes (editors). *Finite-State Language Processing*. The MIT Press, 1997.

[11] M. Vilian and D. Day. Finite-state phrase parsing by rule sequences. In *Proceedings of the 16th International Conference on Computational Linguistics*, 1996.

A Penn Treebank POS tags

Tag	Description	Examples
DT	Determiner	the, this, that
JJ	Adjective	recent, such, high
MD	Modal	should, can, must
NN	Noun	fact, house, weekend
NNP	Noun, proper	Michael, Sydney
POS	Possessive Ending	V
PRP	Pronoun	it, us, you
VB	Verb	be, make, have
VBD	Verb, past tense	was, said, placed
VBN	Verb, past participle	paid, named, made
VBP	Verb, non-3rd person, singular, present	do, argue, have
VBZ	Verb, 3rd person, singular, present	is, has