The Classification of XSLT-Generated Web Documents with Support Vector Machines

Atakan Kurt, Engin Tozal

Fatih University, Computer Eng. Dept.
Istanbul, Turkey
{akurt,engintozal}@fatih.edu.tr

Abstract. XSLT is a transformation language mainly used for converting XML documents to HTML or other formats. Due to its simplicity and flexibility XML has replaced traditional EDI file formats. Most e-business applications store data in XML, convert XML into HTML using XSTL, and publish the HTML documents to the web. Web document classification deals with determining the class of HTML pages. In this paper we argue that the use of XSLT presents an opportunity rather than a challenge to web document classification. We show that it is possible to combine the advantages of both HTML and XML into classification of documents at the XSLT transformation stage, named *XSLT classification*, to attain higher classification rates using Support Vector Machines. The results are both expected and promising. We believe that XSLT classification can become a favorable classification method over HTML or XML classification where XSLT stylesheets are available.

1. Introduction

Data mining has been applied to a much wider spectrum of application domains including web, GIS, multimedia in the last decade facilitated by many remarkable advancements in various branches of information and computing technologies. XML[1] (eXtensible Markup Language) is certainly one of those information technologies that have dramatically impacted many application domains. By bringing structure to unstructured documents, XML practically became a synonym for semi-structured documents in the area of digital libraries or information retrieval.

The widespread use of XML in e-business applications has resulted in the definition of many domain specific XML vocabularies such as ebXML (Electronic Business XML), VRML (Virtual Reality Markup Language), SVG (Scalable Vector Graphics), MathML (Mathematical Markup Language). More recently applications that produce dynamic or static HTML documents have started generating documents/data in XML format first, then converting the XML documents to medium or client specific formats including HTML, XML, text, PDF(Portable Document Format), WML (Wireless Markup Language), etc using XSLT[2] (eXtensible Style

[1] http://www.w3.org/XML/
[2] http://www.w3.org/TR/xslt

Language Transformation), thus simplifying overall software engineering process and cutting down development costs among other obvious advantages.

From the data mining point of view, this new XML-to-HTML-by-XSLT trend seems to complicate things at first look, as the effects of such transformations on the classification of web documents whether HTML or XML were not considered before. A number of web page or site classifcation techniques based on HTML have been introduced into web mining literature [9, 10]. Different approaches are taken in these studies. Text-only approach removes all markups and performs classification based on pure content [7]. HyperText approach considers the markup tags to assign weights of features [8, 9]. Link Analysis approach builds its classification model links among web pages [11, 12]. Semi-Structured Document or XML classification techniques not only consider the textual content or the text but also the markup. Structured vector model [13], tags each word with the enclosing markup to generate a feature set. A different semi-structured document classifier [14] proposes a model based on Bayesian Networks in which the training is done for sub-sections of documents.

In this paper we show that XSLT presents unique opportunities rather than new challenges in web classification. The idea is to combine advantages of HTML and XML with the power of XSL transformations which is presented in more detail in Section 2. The experiments performed on a small data set reveal that XSLT transformation outperforms both HTML and XML classifications using Support Vector Machines, confirming similar findings performed in a previous study [1, 2] using Naïve Bayes.

This paper is organized as follows: In Section 2 we define XSLT and the motivation behind the use of XSLT in web applications. The web/document classification framework based on XSLT used to perform the experiments is briefly introduced in Section 3. We discuss the data set, the experimental setup and a short evaluation of results in Section 4. Conclusions are drawn in Section 5.

2. Background and Motivation

XSLT is part of XSL (Extensible Stylesheet Language) standard that is used to define a set of transformation rules for converting XML documents into different formats as shown in Figure 1. In a way, XSLT is to XML as CSS[1] (Cascading Style Sheets) is to HTML. XSLT is however much more powerful than CSS as it can be used as a full-featured programming language. XSLT Stylesheets (transformation programs written in XSL), as they are called, are themselves written in a specific XML format defined by a DTD[2] (Document Type Definition), and therefore can be processed by other XSLT stylesheets as data. Different stylesheets can be used for different requirements. For example, one stylesheet can be employed to produce WML output for WAP (Wireless Application Protocol) enabled cell phones, a different stylesheet for printer output, and another one for handheld computers with small displays.

[1] http://www.w3.org/Style/CSS/

[2] http://www.w3.org/2001/XMLSchema.dtd

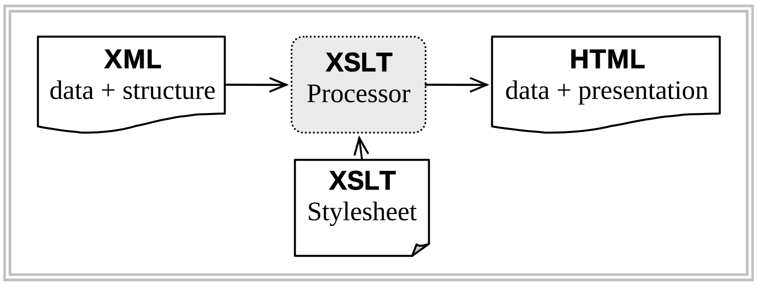

Figure 1 Extensible Stylesheet Language Transformation

As shown in the figure, XML enables us to define the structure and separate the data/content from presentation which is mostly medium/application specific. This advantage over HTML is exploited in the web classification framework used in this study. XSLT stylesheets are executed by XSLT (like Saxon, and Xalan) processors that are available as programming APIs in most languages. XSLT processors are also a part of browsers such as MS Internet Explorer 5.0 and higher versions, and Firefox. This availability opens up many opportunities, since web applications and web documents are mainly accessed via a browser as shown in Figure 2.

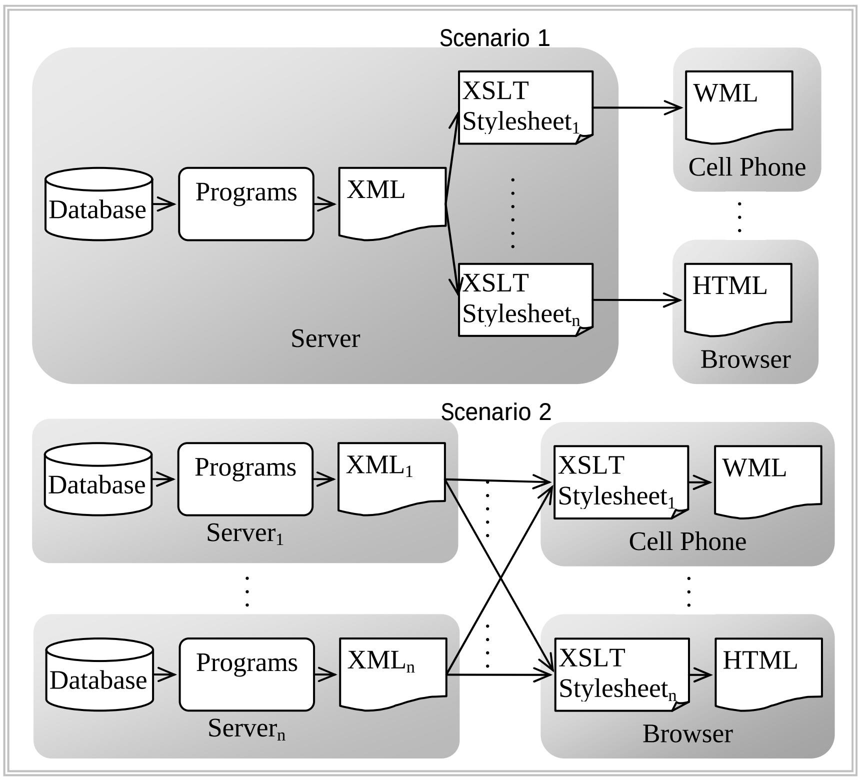

Figure 2 Employment of XSLT in Web Applications

In this figure we show two different web client/server application scenarios. In Scenario 1, an application produces different outputs (WML, HTML, PDF, etc.) for different mediums using a set of stylesheets. Stylesheets are applied to the same XML document which is dynamically produced from data in a database. In this scenario, we assume clients do not have XSLT parsing ability.

In Scenario 2, we illustrate two different situations. Firstly XSLT stylesheets are pushed to the clients. This decreases server load, simplifies server application development, reduces network traffic, and allows clients to choose from a number of stylesheets (not shown in the figure), or use their own stylesheets. Secondly stylesheets can retrieve and combine data from multiple sources which is also possible in Scenario 1.

Figure 3 contains a sample XML document, an XSLT stylesheet applied to this sample document, and the produced HTML output. XSL commands and templates are indicated with the *xsl*: namespace in the stylesheet. XSLT stylesheets are composed of *template*s to be *match*ed from the input. All literals and the results of applying the templates are copied to the output. *Values-of* elements from the input can be *select*ed. Looping, conditional processing, functions are available. Built in filtering and sorting capabilities are parts of the language as well. XPath, XML Path Language is used to selectively address the parts of input XML document. For example *//title* selects all *<title>* elements, while special '*/*' symbol selects the root element from input.

XML

```
<source id=”10021”>
<type>article</type>
<status>APPROVED</status>
<title>XSL</title>
<author>
<fname>John</fname>
<lname>Sabre</lname>
</author>
<comment>Appropriate for
publishing</comment>
</source>
```

HTML

```
<html><head>
<meta name="keywords"
content="article, xml,
xslt, xsl, web"/>
<title>XSL ERA</title>
</head><body>
<h1>XSL</h1>
<h2>John Sabre</h2>
Articles about XML, XSL,
and new web technologies…
</body></html>
```

XSLT

```
<xsl:stylesheet version = '1.0'
     xmlns:xsl='http://../Transform'>
<xsl:template match="/">
<html><head><meta name="keywords"
content="article, xml, xslt,
xsl"/><title>XSL
Era</title></head><body>
<h1><xsl:value-of
select="//title"/></h1>
<h2><xsl:value-of
select="//author/fname"/> <xsl:value-
of select="//author/lname"/></h2>
Articles about XML, XSL, and new web
technologies…
</body></html>
</xsl:template>
</xsl:stylesheet>
```

Figure 3 Sample XML, XSLT and HTML Documents

3. The XSLT Classification Framework

XSLT classification is a hybrid classification technique that exploits both structural markup in XML and presentational markup features in HTML. The logical system layout is depicted in Figure 4. In this figure HTML is shown in gray, since we do not work on HTML documents directly. The XSLT classification uses data (content), structure in XML, and other heuristics from HTML to improve classification rates.

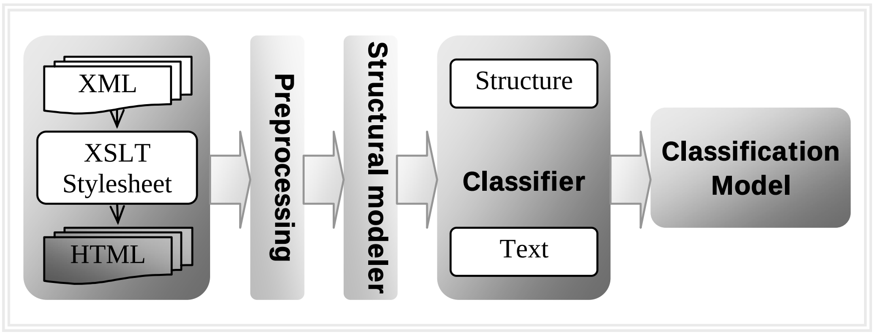

Figure 4 XSLT Classification Overview

We think that XSLT classification produces better results than HTML classification, because XML documents contain structural information of nesting and content specific markup vocabulary not present in HTML.

We believe that XSLT classification is better than XML classification too, because:

- An XML document usually contains meta-data that is not related to the subject of the document but used for other purposes by the generators of document. Usually that data is not presented to end-user, and should be omitted in the classification process. Elements such as *id, type, status* are examples of meta-data in Figure 3. We can eliminate meta portions of XML document using XSLT classification
- There may be some literals valuable in classification presented to the user as part of HTML but not part of the XML document such as sentence *"Articles about XML, XSL, and new web technologies…"* as shown in Figure 3. This type of data can be processed with XSLT easily.
- Some important data that is embedded in HTML tags like <title>, <img>, <meta> but not present in XML should be captured with XSLT.
- Sometimes only a part of an XML document may be relevant for the end-user or relevant to the subject of the HTML document. In this case, the irrelevant parts of the XML should be discarded all together in the classification.
- It may be the case that an HTML output could be combination of a set XML documents. It would be wiser to consider the output HTML, instead of input documents in classification which is the case with XSLT classification.

The architecture of web classification framework based on XSLT is shown in Figure 5. The framework consists of three modules; *Preprocessor, Semi-Structured Document Modeler, and Classifiers*. The system accepts a set of XML documents and an XSLT stylesheet to transform them in HTML or other formats as input. The output of the framework is a classification model for the given training set using support vector machines which will be briefly explained below. The framework can be used to classify documents in both Scenario 1 and Scenario 2.

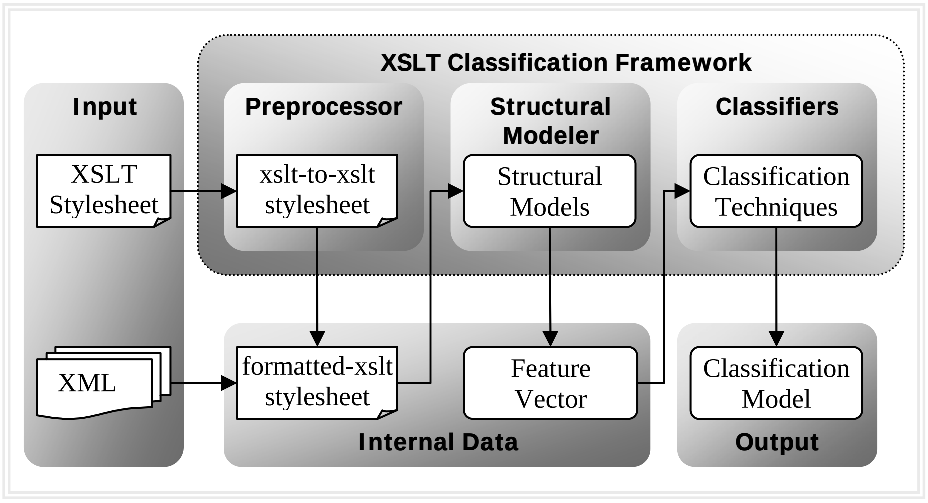

Figure 5 XSLT Classification Framework Architecture

The original XSLT document is passed to the preprocessor in which an XSLT-to-XSLT stylesheet is applied to produce a new XSLT document called *formatted-XSLT stylesheet.* The formatted-XSLT is an XSLT stylesheet that contains transformation rules in the original XSLT, at the same time, is able to produce appropriate outputs of XML documents for the structural modeler. These *formatted-XML* documents (not shown in the figure) generated by applying formatted-XSLT to the XML documents are then fed to the structural modeler which applies one of the structural models defined in the study [1]. The modeler produces a feature vector for the term frequency vectors (not shown) used later in the classification step. A feature vector usually contains all unique words mostly prefixed with the XML tags in which they reside in the original XML document. Term frequency vectors are generated for all formatted-XML documents and passed as a training set to the classifier. The classifier contains a number of classification algorithms in Weka. SVM is used in this study. The classifier creates a model based on the training set. This model is used to classify new documents. In the following sub-sections we briefly discuss components of the framework.

3.1 Preprocessor

In the preprocessing step, an XSLT-to-XSLT stylesheet is applied to the original XSLT stylesheet to generate *formatted-XSLT stylesheet*. The XSLT-to-XSLT stylesheet simply traverses each element of the original XSLT document and does the following to produce the XSLT stylesheet which is referred to as result tree below.

The *ancestor-or-self* refers to creating a string by concatenating all the element names from the root to the innermost element separated with '-_-' as shown in Figure 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<document-root>
article xml xslt xsl XSL Era <source-_-title>XSL</source-_-
title>
    <source-_-author-_-fname>John</source-_-author-_-fname>
    <source-_-author-_-lname>Sabre</source-_-author-_-lname>
Articles about XML XSL and new web technologies
</document-root>
```

Figure 6 Sample Formatted-XML Document

- If the current node is an *xsl:element* node whose name is not an HTML tag, insert it into result tree and process its child-nodes.
- If the current node is an *xsl:vlaue-of* element, then insert the value referred by *select* attribute into the result tree with all its *ancestor-or-self* hierarchy.
- If the current node is an *xsl:text* element, then insert the text into the result tree with all its *ancestor-or-self* hierarchy.
- If the current node is any other XSLT element -*xsl:variable, xsl:param, xsl:with-param, xsl:if, xsl:when, xsl:choose, xsl:otherwise, xsl:copy, xsl:copy-of, xsl:sort, xsl:for-each*- put it directly into the result tree and process its children.
- If the current node is an HTML *title*, *meta, img/alt* tag, insert its content into the result tree.
- If any other string literals, copy them into the result tree.

3.2 Semi-Structured Document Modeler

Semi-Structured document modeler generates a feature vector from the set of formatted-XML documents. Later the frequencies of each unique feature or word are placed into term frequency vectors for each formatted XML document using a structural representation model.

There are a number of alternatives (as explained in [1]) of incorporating structural information into document classification such as prefixing the word with the innermost enclosing tag or all inclosing tags etc. The strength of each alternative model is affected both by how the structure is represented in the term frequency vectors and by the variations in element, removal or insertion of inter elements, and the swap of elements in the document. We show how a document is represented in feature vectors in this study. The example below is based on XML document given in Figure 3. `source, author, fname` are tags, *`John`* is a text content.

Table 1 Sample Term Frequency Vector

Feature Vector	Term Frequency Vectors			
	D_1	D_2	...	D_n
...				
source	8	6		4
author	4	2		0
source.author	3	1		0
fname	3	1		1
source.fname	2	2		2
author.fname	2	3		0
john	*1*	0		*1*
source.*john*	1	0		1
author.*john*	1	0		0
source.*john*	1	0		1
. . .				

A feature for term frequency vector is created as follows: each unique word or element is a feature. Each word and element is prefixed with each of its ancestor separately to create new features. Furthermore each of the ancestor elements is prefixed with their ancestor elements to create new features. Table 1 shows term frequency vectors for the *"<source-_-author-_-fname>John</source-_-author-_-fname>"* fragment of formatted-XML document given in Figure 6. Values shown in the table are not actual values.

Although the structure is captured in a loose manner (i.e. we do not capture ancestor hierarchy in a strict manner), the complete document hierarchy is captured. This representation is resistant to the structural alterations to some degree.

3.3 Classifier

Support Vector Machines and Naïve Bayes are two well known and robust classification techniques used in data mining. The experiments performed in this study are based on both to ensure the correctness of empirical results.

3.3.1 Naïve Bayes

Given a document database $D=\{d_1,d_2,...,d_n\}$ and a set of categories $C=\{c1,c2,...,cm\}$ the classification problem is to define a mapping $f : D \rightarrow C$ where each d_i is assigned to one category. A category contains those documents mapped to it; that is $c_j = \{d_i \mid f(d_i) = c_j, 1 \leq i \leq n$ and $d_i \in D, cj \in C\}$.

In Naïve Bayes IR text classification technique [15, 16]; given the set of categories $c=\{c_1,c_2,...,c_m\}$, and a set of documents $d=\{d_1,d_2,...,d_k\}$ a document d is represented as an n dimensional feature vector $d=\{w_1,w_2,....w_n\}$ where the occurrence frequency of i^{th} word is kept as value of a feature w_i, $1 \leq i \leq n$. An estimation of conditional class probability of document d_j belongs to category c_i is obtained by formula

$$P(c_i \mid d_j) = \frac{P(c_i) * P(d_j \mid c_i)}{P(d_j)}, \; 1 \leq i \leq m, \text{ and } 1 \leq j \leq k$$

$P(d_j)$ which is prior document probability is same for each class i so there is no need to calculate. Prior class probabilities $P(c_i)$ for each class i can be estimated from the frequencies of documents belonging to class c_i in the training data. Estimating the probability of the feature vector document d_j given the class c_i, $P(d_j|c_i)$ is expensive to compute, due to the fact that; a feature w_r can take a big number of values. In order to make the calculation simple each feature is assumed to be independent, this is the core of Naïve Bayes Classifier model. So $P(d_j|c_i)$ is calculated under Naïve Bayes multinomial model [17] by formula;

$$P(d_j \mid c_i) = \prod_{w \in d_j} P(w \mid c_i)^{f(w,d_j)}$$

$f(w,d_j)$ is the number of occurrences of word w in document d_j.

3.3.2 Support Vector Machines

Support Vector Machines [[5] are well known and powerful classifiers whose theory is developed by V. Vapnik. Simply an SVM considers the training data as vectors in Euclidian space and tries to come up with the optimal separating hyper-surface in the space. The optimal separating hyper-surface is the one with the largest distance between itself and closest instance of each class. In its simplest form, linear and separable, an SVM [[3] is a hyperplane that distinguishes a set of positive instancess from negative instances as shown in Figure 7.

Given a set of labeled training examples $\{(x_1,y_1), (x_2,y_2), (x_3,y_3),..., (x_N,y_N)\}$ where example $x_i \in R^d$ is a d-dimensional term frequency vector representing an XML document, and $y_i \in \{-1,+1\}$ is the class of x_i.

$$u = \vec{w}.\vec{x} - b, \;\; \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

u is the classification output of the machine, $x_i s$ are the training set, x is new instance to be classified, $y_i \in \{-1,+1\}$ is output class for x, b is a threshold, α_i are adjustable parameters of the hyperplane. The objective of training of SVM is to find best the α_i which is an optimization problem solved by the algorithm.

[3, 4], and [6] show that SVM perform better than many other text classification algorithms.

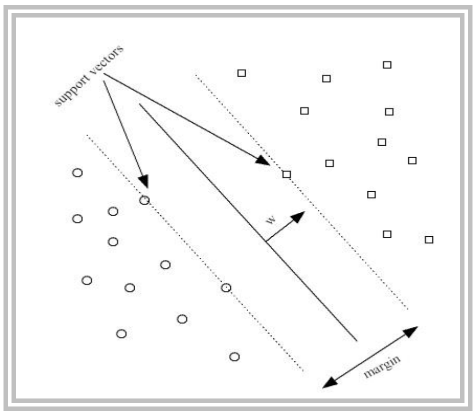

Figure 7 SVM for Linear and Separable Case

4. Experiments, Results, and Evaluation

We performed a set of experiments to compare the classification rates for HTML, XML, and XSLT classifications using the framework described above. The framework was implemented in Java and XSLT. We utilized the Saxon-B 8.4[1] library in implementation. We used Weka for the classification algorithms.

2/3 of documents are used for training and 1/3 for testing with 10-fold cross validation to improve the reliability. The element/attribute names and the words in text are stemmed (taking the root of a word) in all experiments, as it is a common practice in text mining.

4.2 Dataset

Current dataset repositories on the web do not provide a proper dataset for the XSLT classification. We generated XML/XSLT version of web pages from 20 different sites belonging to 4 different categories; *Automotive, Movie, Software, News & Reference*. The sites in *News & Reference* category contain news and articles about movies, automobiles and software health and literature. This should result in a more difficult classification task. The data set can be downloaded[2] freely. 100 XML documents were generated from the web sites. These documents are evenly distributed among categories. XML documents are created in a way that they have various structures, element and attribute names, and nesting to mimic that they are generated by different people/applications. An XSLT stylesheet producing exactly the same presentation with all links, images, embedded objects, literal strings and non-printable data like meta, style, script tags etc. of actual HTML page is generated for each web site. When the XSLT is applied to the XML document, it combines static content with dynamic

[1] http://www.saxonica.com
[2] http://www.fatih.edu.tr/~engin

content. Static content includes menus, scripts, headings, and other similar material. Dynamic content is the data retrieved from the XML files. By combining the the two contents, an HTML or XHTML page is created.

Table 2 APR. TFV Size

Data Type	# Features
XML	~37000
HTML	~7000
XSLT	~34000

Table 3 Dataset Properties

Data Set	# Classes	# Sites	# Documents
1	4	17	91
2	4	14	75
3	3	13	70
4	3	13	60

Since data size is quite limited, we have created 4 different versions of the data set by excluding either some categories or some web sites or both randomly from the original data set. Characteristics of data sets are shown in Table 2 and Table 3. We conducted experiments on these data set using SVM and Naive Bayes to compare the results with the previous study [2] which used only the Naïve Bayes. Experience dictates that not all classification algorithms do well with a certain type of data. Data mining is an experimental science. It is also a common practice, called voting, to apply a number of different techniques to classify a new instance by choosing the category preferred by the highest number of techniques. For these reasons, we think that it is necessary to conduct further experiments with different techniques and datasets on XSLT classification as explained in the next section.

All presentation markups are removed while generating feature set for HTML documents. Yet contents of *meta, title, anchor* and alternative name for *img* tags are included into the feature set. The structural modeling technique explained in Section 3.2 is not only used to generate feature set for XSLT documents but also for XML documents. Moreover, %2-%4 noise is introduced into XML documents instead of XML meta data.

4.4 Results and Evaluation

The experimental results are shown in tabular form per data set in Table 4 and average classifications over all datasets are depicted in Figure 8. In general XSLT classification yields considerably higher accuracy rates than both HTML and XML classification, while XML classification produced slightly better accuracy rates than HTML classification. The results reveal that both SVM and Naïve Bayes deliver similar rates confirming each other’s output.

In all data sets except the last one, the XSLT classification performs better than the other two, while XML classification yields better scores than HTML classification. Noticeably Naïve Bayes outperforms SVM on both XSLT and XML data. Since the figures are close and data sets are small, between Bayes and SVM it s not possible determine which one is better. However there seems to be marked difference between the two methods on HTML data. As shown in Table 2, representing structure in XML and XSLT classifications results in much larger term frequency vectors than those of

HTML classification. Since XML and XSLT produce higher classification rates, this is a trade-off between accuracy and space. However an empirical threshold value can be used to reduce the term frequency vector size in XML and XSLT classifications. As shown in Table 4 SVM takes longer in building classification model compared to Naïve Bayes. The time requirement of SVM is much more than that of Naïve Bayes'.

Table 4 Accuracy Rates of SVM and NB

Data Set	Type	Bayes	SVM	Bayes Time	SVM Time
1	HTML	94.44	96.6	20	490
	XML	97.7	97.7	60	1000
	XSLT	100	100	50	950
2	HTML	93	96	30	2090
	XML	97.3	97.3	910	890
	XSLT	100	100	50	1630
3	HTML	92.8	95	20	360
	XML	97.14	95.7	50	660
	XSLT	100	100	50	590
4	HTML	95	95	20	270
	XML	95	95	40	880
	XSLT	100	96.6	50	890

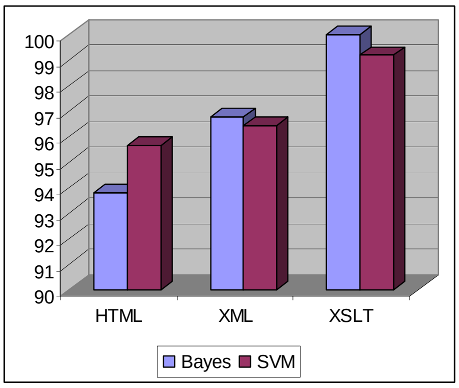

Figure 8 Average Accuracy Rates

5. Conclusion

XSLT is used in more and more applications because of the ease, power and flexibility it offers in software development. Web applications producing output using XML/XSLT technology allows three types of classification options; classification at the source (XML classification), classification at the destination (HTML classification), and a new alternative: classification at the point of XSLT transformation. We have explored the third option for classifying web pages and showed that it is not only viable but also a preferable approach over the others as it takes advantages of both approaches. This technique is able to combine both the source and the destination document for better classification. More specifically, it is able utilize both structural data in XML and relevant data in HTML using the transformation rules in XSLT stylesheets. As a result a technique with a considerably higher classification rate is obtained.

We implemented a framework that incorporates the XSLT classification in a practical manner to classify web pages. In this framework different structural models and alternative classifiers can be combined to classify documents generated by XSLT Stylesheets.

Even though many e-business applications are using XSLT internally to generate and share XML/HTML documents, applications that rely on client side XSLT is rare.

Although there are browsers with built-in XSLT processor, other types of clients such as cell phone, PDAs, TV sets do not have widespread XSLT support at present. This situation restricts applications to server side transformations in todays applications. With the availability of larger public datasets in the future, further experiments can be performed.

References

[1] Engin Tozal, “Classification Using XSLT” MS Thesis, Computer Engineering, Fatih University, Istanbul, September 2005.

[2] Atakan Kurt, Engin Tozal, "A Web Classification Framework Based on XSLT" ADWeb 2006 Lecture Notes in Computer Science (LNCS) 3842, pp. 86 – 96, 2006.

[3] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, “Inductive learning algorithms and representations for text categorization”, In Proceedings of the seventh international conference on Information and knowledge management, pages 148--155. ACM Press, 1998.

[4] Thorsten Joachims, "Text categorization with support vector machines: learning with many relevant features", Proceedings of {ECML}-98, 10th European Conference on Machine Learning, 1998.

[5] V. N. Vapnik, The Nature of Statistical Learning Theory”, Springer, 2nd edition, 1999.

[6] A. Basu, C. Watters, and M. Shepherd, ”Support Vector Machines for Text Categorization”, Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.

[7] Dunja Mladenic, “Turning Yahoo to Automatic Web-Page Classifier”, European Conference on Artificial Intelligence, 1998

[8] F. Esposto, D. Malerba, L. D. Pace, and P. Leo. “A machine learning apporach to web mining”, In Proc. Of the 6th Congress of the Italian Association for Artificial Intelligence, 1999

[9] A. Sun and E. Lim and W. Ng, “Web classification using support vector machine”, Proceedings of the fourth international workshop on Web information and data management. ACM Press, 2002

[10] Arul Prakash Asirvatham, Kranthi Kumar Ravi, “Web Page Classification based on Document Structure”, 2001

[11] H.-J. Oh, S. H. Myaeng, and M.-H. Lee, “A practical hypertext categorization method using links and incrementally available class information”, Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval, 2000

[12] Soumen Chakrabarti and Byron E. Dom and Piotr Indyk, “Enhanced hypertext categorization using hyperlinks”, Proceedings of {SIGMOD}-98, {ACM} International Conference on Management of Data, 1998

[13] Jeonghee Yi and Neel Sundaresan, ”A classifier for semi-structured documents”, Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000.

[14] Ludovic Denoyer and Patrick Gallinari, ”Bayesian network model for semi-structured document classification”, Information Processing and Management, Volume 40, Issue 5, 2004.

[15] David D. Lewis, “Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval” Lecture Notes in Computer Science; Vol. 1398, 1998.

[16] Irina Rish, "An empirical study of the naive Bayes classifier", IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, 2001.

[17] Andrew McCallum and K. Nigam, “A comparision of event models for naive bayes text classification”, AAAI-98 Workshop on Learning for Text Categorization, 1998.