A BOTTOM-UP APPROACH FOR XML DOCUMENT CLASSIFICATION

JUNWEI WU

A Bottom-up Approach for XML Document Classification

by

© Junwei Wu

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

Jun. 2009

St. John's Newfoundland

Abstract

Extensible Markup Language (XML) is a simple and flexible text format derived from Standard Generalized Markup Language (SGML) [1]. It has been widely accepted as a crucial component of many information retrieval related applications, such as XML databases, web services, etc. One of the reasons for its wide acceptance is its customized format during data transmission or storage. Classification is an important data mining task that aims to assign unknown objects to classes that best characterize them. In this thesis, we propose a method to classify XML documents under the assumption that they do not have a common schema that may or may not be available, which is closer to the real cases. Our method is similarity-based. Its main characteristic is its way to handle the roles played by texts and the structural information. Unlike most existing methods, we use a bottom-up approach, i.e., we start from the text first, and then embed the structural information. This is based on the observation that in XML documents with diversified tag structures, the most informative information is carried by the terms in the texts. Our experiments show that this strategy can achieve a better performance than the existing methods for documents from sources that exhibit heterogeneous structures.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Jian Tang, without whose continuous support over the past three years, I could never have achieved this goal. His inspiring suggestions, enlightening discussions, as well as serious thinking have greatly helped to improve this thesis.

I am also grateful for the providers of the data set [14] used in this thesis, and for the thousands of individuals who have been dedicated to the development of the robust and freely available software tools, such as Eclipse, Weka, LaTeX, and so on.

Finally, I wish to express my deep gratitude to my beloved ones: my parents and my wife. The steady support and encouragement from them has been, and continues to be, a source of inspiration for me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the current information age, text file has become an important information source. Text classification, which aims to assign unknown objects into predefined classes that characterize the objects, gains more and more focus. Although text classification dates back to the 1960s, automated text classification began to attract attention around two decades ago. Until the late 1980s, the mainstream of the applicable application on text classification is based on knowledge engineering (KE) [17], which makes prediction based on a set of rules set manually by the experts in this field in advance. Since the 1990s, text classification has been an active area in the data mining field [15][17][27]. A machine learning method is often used to automatically build a classifier by learning the information in the training samples. The advantage of this method is the saving on the domain experts' manual effort for one particular classifier system as well as the comparable prediction accuracy to the previous KE based system.

Semi-structured documents is a type of text document, which usually follows some

standards, such as HTML, email, and XML (Extensible Markup Language) [1]. Unlike the first two document formats, the standard of XML is more open and flexible. It allows the users to define the structures themselves to some degree. This openness and flexibility of XML provide convenience to the users. Nonetheless, it introduces some uncertainties of the structures at the same time.

Like the task of text classification (TC, a.k.a text categorization, or topic spotting [17]), semi-structured document classification is used to predict the class label of an unlabeled text document based on learning from a group of labeled training samples. With the rising number of semi-structured documents, it is increasingly important to support quick and effective information processing, such as retrieval, search and filtering. Among other things, grouping them into correct categories first will greatly facilitate such a process. For instance, at news websites there might be several categories of news in XML format, such as Business, Sports, Health, Tech &Science, and Entertainment. Before being published, it is desirable to automatically classify the high volume of news into their respective categories.

There are several challenges associated with automated text categorization: an appropriate document representation, an appropriate classifier function to obtain good generalization, avoiding over-fitting and an appropriate dimensionality reduction method to handle algorithmic issues [7].

However, employing the conventional text classifiers directly on semi-structured document classification does not perform well because they are designed for non-structured data [27]. A crucial issue for semi-structured document classification, though, is how to exploit their structures [5][8][9][15][20][27][29]. For HTML or email documents, this is relatively easy, since the tags appearing in these documents nor-

mally follow some fixed standards. Users cannot add self-defined elements that are not defined in the standards. This enables the analysts to take advantages of the limited and predefined element tags to build highly accurate classifiers [13][18][19]. Unlike HTML or email, however, all the elements in XML documents can be predefined by the authors themselves via a DTD (Document Type Definition) file. During the learning process, if all the samples in the training set follow a common DTD, then the process of hypothesis formulation (i.e., classifiers) can be facilitated [8][9][15]. However, when the XML documents come from different sources, it is likely that they do not share a common DTD, and, furthermore, these DTDs are not even available.

In this thesis, we propose a bottom-up scheme for the classification of XML documents that may come from different sources and can follow different DTDs. These documents have similar but not identical structures, such as elements, attributes, and their relationships. We borrow the definition of "similar DTDs" from the homogeneity-hypothesis in [8], i.e., "similar DTDs carry similar structural information (documents are more or less of the same type), but that tags may have different names, some may be missing and some may be added". This assumption is closer to the real world data. To simplify the problem, in this thesis, every document belongs to one and only one class. Our method is similarity-based. It differs from the existing schemes in that it is based on a bottom-up approach, i.e., we start from the text first, and then embed the structural information. This is based on the observation that in XML documents with diversified tag structures, the most informative information is carried by the terms in the texts. Our experiments show that this strategy can achieve a better performance than existing methods for documents from sources that exhibit heterogeneous structures.

The rest of this thesis is organized as follows. In Chapter 2, we survey on related work. We cover text classification briefly focus more on the semi-document classification. In Chapter 3, we give the detail of our method. In Chapter 4, we present the experiments as well as the experimental results. In Chapter 5, we draw our conclusion.

Chapter 2

Related work

2.1 Text classificaton

In [17], Sebastiani compares a group of state-of-the-art approaches to text classification. He discusses in detail three problems: document representation, classifier construction and classifier evaluation. The conclusion is that constructing a data structure that can represent the document, and constructing a classifier that can be used to predict the class label of a document with high accuracy, are the key objective tasks in text classification.

Document representation is a key point in the document mining task. The better the representation is adopted, the less information is lost, thus better results should be achieved. To represent a document, feature selection is usually required due to the high dimensionality of the text document. Yang and Pedersen evaluate 5 different aggressive dimensionality reduction methods [26]. Rogati and Yang report a controlled study on a large number of filter feature selection methods for text classification [16].

Dasgupta et al. discuss the feature selection methods for text classification in [7]. Fragoudis et al. present their approach that deals with both the problems of instance selection as well as feature selection for reducing the amount and complexity of data in text classification [12].

Classifier construction concerns how to choose the classifiers. Eyheramendy et al. empirically compare the performance of 4 probabilistic models for text classification[10]. They find that, in general, relaxing the "Naive Bayes" assumption in the 4 models does not change the performance of the classifier. Sebastiani gives a list of classifiers in detail. He also collects the published experimental results for some considerations on the comparative performance of the TC methods involved. There is no sufficient evidence showing which one is always the best classifier in different circumstances[17].

"The experimental evaluation of a classifier usually measures its effectiveness (rather than its efficiency)" [17]. In the binary classification, which is a task to predict whether a test document belongs to a predefined class or not, the precision and recall as well as the F1 measure are widely used to evaluate a classification system [7][10][12]. Generally, however, in the multi-class-single-label classification (i.e., multiple class labels exist and every document belongs and only belongs to one class), researchers provide an accuracy or error rate [14][27][29].

2.2 Semi-structured text classification

In the semi-structured document classification field, many researchers have proposed different classification methods to do the semi-structured documents classification task [5][6][8][9][14]

[15][20][21][23][25][27][29]. Most utilize the combination of the content and embedded structure [5][8][9][15][20][25][27][29].

Some researchers utilize the limited and fixed structures in the current email or HTML standards [6][13]. However, this does not apply to classification on XML documents with different structures. Researchers propose different ways to utilize the structural information within the XML documents.

Yi and Sundaresan describe a novel text classifier that can effectively cope with structured documents [27]. It is one of the earliest attempts to do semi-structured document classification with the consideration of the information included in the embedded structure as well as the textual information in the leaf nodes belonging to the XML documents. In their methods, to keep the information in the structure, firstly, a structured vector model is developed, which represents a document with a structured vector. The elements of the vector can either be terms or nested structured vectors. Secondly, the well known Bernoulli Document Generation Model is extended to a probabilistic document classification model for structured vectors. "This is the first classification system that feeds both textual and structural features into a general statistical model in order to classify semi-structured documents" [27]. This method extends the Bernoulli Document Generation Model by filling in the information collected via counting the terms along the paths of a document. The documents belonging to a class do not have to conform to a specific schema. However, it makes a weaker assumption than the conventional classifier. Other than requiring independence for the entire set of terms of the document, it instead requires the assumption to be held only in the structure nodes to which the terms belong. Moreover, it assumes that different levels of the paths in a document are independent of each

other. This classifier is also a document ranking classifier [17], i.e. to predict the class of an unlabeled document by ranking the posterior probability $P(C|d)$, where d is the given unlabeled document, C is the respective class. The class that maximizes the posterior probability is chosen as the class label.

A method based on the frequent sub-structures contained in XML documents is proposed in [29]. This scheme, however, implicitly assumes that documents in the same class share sub-structures in common, while documents from different classes do not. Zaki and Aggarwal propose this rule based method by using frequent discriminatory substructures within XML documents. The core part of this method is to construct the structural rules in order to perform the classification task. In the training phrase, the substructures that are closely associated with one certain class are to determined because they believe that "the presence of a particular kind of structural pattern in an XML document is related to its likelihood of belonging to a particular class" [29]. In order to do so, XMiner, which mines pertinent structures for multiple classes simultaneously, is extended from TreeMiner [28]. It accepts as input a list of minimum support thresholds for each class, and outputs a set of frequent rules for each class, respectively. In the process of mining the structures, a special vertical tree representation for fast support counting, scope list, is proposed, moreover, join operation, a special operation applying to scope list is also proposed to help to find the frequent sub-structures. Rule support and rule strengths are defined to measure the qualification of the rules. When the satisfied frequent substructures are found, based on them, rules that are less predictive will be removed. The qualified rules for every class will be sorted according to the strengths and support, respectively. Moreover, a default class will be determined so that, when no rule can cover an example tree,

the default class will be appointed to that document. But, the issue is that it is not easy to define the rule support and rule strengths when facing different data sets. In addition, this method is suitable for data sets that do have some special characteristic substructures in one class against any other classes. If the data sets do not have these substructures in each of the classes, or when this substructures are hard to be mined out, the classification performance will be affected.

Denoyer and Gallinari present 2 papers on semi-structured/XML document classification/categorization [8][9]. Both of them model the XML document as a Bayesian Network (BN). It is assumed that the children nodes are only dependent on their parent nodes, respectively. It requires the assumption that the textual words are independently held in the structure node it belongs to, which is similar to the one in [27]. The information included in the XML document is divided into 3 parts: logical structure information, label information and the textual information. As the textual information held in one node is assumed to be independent, a leaf node that is made up of textual contents can be divided into several independent textual one-word nodes, all of which are the children nodes of the structural node that contains this leaf node. In the first part of the [8], it is assumed that all the XML documents share a common DTD so that all the parent-child relationships can be estimated from the training set, i.e., the probability of a child node given the known parent node can be estimated in advance. According to the characteristic of the BN model as well as the Baysian rule, the posterial probability $P(C|d)$ can be estimated for a test document. The values for different classes will be ranked, and the highest one will be picked as the class label.

In the second part of [8], the authors extend the method to the unknown DTD

XML document classification. That is, the XML documents involved do not necessarily share a common DTD. In their method, all the node labels in the test document will be ignored, and it tries to fill in the node labels based on the structures of the test document as well as the training set. After the node labels have been filled, it can do the calculation as the first part does. However, this can lose the node label information and it is a time-consuming process.

In [9], the first part is same as [8], however, in order to improve the discriminative abilities of the generative model, the authors employ the Fisher score for each document d and each class c, thus, a vector corresponding to the document representation in the Fisher space can be formed. Then the SVM (Support Vector Machine) classifier is used to do the binary classification.

Bratko and Filipic investigate methods exploiting structural information in the semi-structured documents when doing the document classification by the Naive Bayes classifier [5]. The authors try different document modeling methods to estimate the posterior probability $P(d|c)$. Component tagging, which treats word occurrences in different document components as different features, is a common approach to model the semi-structured document. Another one is "component splitting", which is to train as many classifiers as there are structural components. The final classification is a combination of the results of all the classifiers. Bratko and Filipic also employ different component smoothing methods to solve the data sparseness and out-of-vocabulary (OOV) issues. The experiment shows that the conventional flat Bag-of-Words representation for the text classification is outperformed on every single data set; this supports the argument that the utilization of the structural information in the semi-structured documents does help to improve the performance of

the document classification.

Theobald et al. propose a methodology on how to automatically classify schema-less XML data into a user-defined topic directory [20]. The normal ways to use only text terms and their frequencies as features for automatic classification of text documents often lead to unsatisfied results because of the noise and different styles of the document authors. The key lies in a more precise document characterization for semi-structure data by exploiting its structure and annotation. The authors' main focus is on constructing appropriate feature spaces on which a classifier operates. The XML twigs and tag paths combined with text terms are studied as extended candidate features. The Mutual Information (MI) between one particular candidate feature of the documents and one particular topic are computed based on the training set. Thus, the relevance of the candidate feature and the specific topic can be measured. As well, the top m features will be selected based on the MI values. The term frequency (tf) and the inverse document frequency (idf) statistics for candidate features will be computed and used as the weight of the features. The highest weighted features are those that are frequent in one document but infrequent across the corpus. Rather than using the tags and terms directly as a feature of an XML document, the WordNet thesaurus is employed as the ontological background to reduce the semantic ambiguity of one word under different circumstances or same meaning conveyed by different words. When the feature space is generated, the SVM is chosen as the classifier. It is an innovative way to measure the relevance between a selected feature and a class in this work. However, the pattern or patterns selected for the candidate features are fixed beforehand, thus, the results can be greatly affected by the structures in the XML documents. Moreover, in the MI formula of this work, the binary value that

stands for the existance of the candidate feature cannot tell the quantitive difference of this candidate feature in different documents from different classes, when all of those documents contain that candidate feature.

We believe that the bottom-up approach can reduce, to a certain level, the sensitivity to the difference on the XML structures in the feature selection. In [22], we describe the basic idea and the advantages of bottom-up approach for classifying XML documents.

Marteau et al. propose a model based on Bayesian classification for the semi-structured data classification task [15]. Their method is also an attempt to exploit the structural knowledge in the semi-structured document when doing the categorization. It is also a document ranking method similar to [8][9][27], which means the classifier is based on the probability of a test document in every predefined class, respectively. The class maximizing the probability will be chosen. The main task is to estimate the posterior probability $P(d|c)$, where d is the test document and c is the candidate class label. The XML document is modeled as a DOM (Document Object Model) tree. There are several hypotheses requested: 1) the orderless occurrence of the sub-trees (i.e., the occurrence order of the sub-trees is not important); 2) the conditional independence of the sibling sub-trees given the root node and the category; 3) the conditional independence of characteristics included in one leaf node. Particularly, it is proposed to set a weight for the node that has a textual element attached. The weight proposed can be the proportion of the cardinal of the vocabulary associated to the node for this candidate category and the size of the textual element attached to this node inside the test document. In other words, relatively, the more vocabulary a node associates with in one certain class, the more important this node is for this

particular class. Just as the authors state, "Thus, the weight associated to node n in document d is a kind of quality criteria that measures the vocabulary coverage of node n for document d" [15]. Their experimental result shows that the document structure plays an important role for the classification. Their model attains the same level accuracy as the SVM model applied on flat text or split textual components, given an ad hoc heuristic independent from the categories.

Xing et al. propose an algorithm for computing the edit distance between a document and schemas [24] and to classify the XML documents based on those edit distances [25]. Firstly, the task is to extract the schema from the XML documents. Actually, here, the authors assume that there exists a schema in a collection of XML documents, $d_1, d_2, ..., d_n$, such that those XML documents are document instances that can be generated by this schema. They also assume that XML documents from different classes have different schemas. An XML document will be represented in Vector Space Model as a $|T|$ dimensional vector where $|T|$ is the number of terms. $|d|$ documents can form a $|T| \times |d|$ term-by-document matrix. By adopting the Latent Semantic Indexing (LSI) and Singular Value Decomposition (SVD), the number of dimensions can be reduced. After the m schemas are ready, the distance between an XML document and those m schemas will be computed to form an m dimensional vector $< D_1, D_2, .., D_m >$, where D_k is the distance between this document and the schema of the class k ($1 \leq k \leq m$). Once each of the documents can be represented in a vector form, it is easy to input these vectors to a classifier and train the classifier. In their study, the distance between a document and a schema is top-down edit distance [24] and the classifier SVM is used. However, the assumption that there is a common schema that can be extracted from one class cannot always be guaranteed.

Kurt and Tozal propose a method to do the classification by exploiting the extra information included in the XSLT documents that are used to convert the corresponding XML documents into HTML or other documents [14]. They argue that the use of XSLT presents an opportunity rather than a challenge to web document classification. The framework consists of 3 modules: Preprocessor, Semi-structured Document Modeler and Classifier. The original XSLT documents are transferred into formatted XSLT stylesheets which can be used to transfer the original XML documents into the corresponding formatted XML documents. Then these formatted XML documents can be output to the structural modeler to generate the feature vectors for every XML document. After generating the vector representation of the XML documents in both the training set and the testing set, SVM is used as a classifier to do the classification task on those vectors. This method takes into account all the element names from the root to the innermost element by concatenating all of them with a specific character in the formatted XML documents. As well, in the processing of the structural modeler stage, each unique word or element is considered as a feature, whose frequency will be counted and filled in the corresponding position as the value for that attribute in the document vector. Similarly, the authors mention, "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" [14]. Their experiment shows that, in general, XSLT classification yields considerably higher accuracy rates than both HTML and XML classification. XML classification performs slightly better than the HTML classification. However, the scheme makes use of the information contained in XSLT files that are not always available. Due to its dependency on the availability of XSLT files, this method's applicability is limited.

Co-training is a strategy for using unlabeled data which has two seperate and redundant views [23]. Wu [23] presents an algorithm of using Co-training with decision tree to handle the labeled - unlabeled problem of XML classification. The method reportedly can solve the problem that a large quantity of labeled samples is needed to learn accurately in the supervised learning. The intuition of the Co-training is that it may be easier for a learner to identify an example and this example may provide the useful information to the other learner. The basic idea behind the co-training framework is to exploit the compatibility between different views on an example. Predicate rewrite systems are used to generate sub-feature spaces for one feature space. Unlike the other Co-training in which the views generated are fixed and set up before the process starts, the predicate rewrite systems in this method can be easily modified to change the views generated. Nevertheless, this also needs human interference every time when classifying different data sets. Moreover, how to choose the predicate rewrite system is a key point and it is still an open question to generate conditionally independent views for an individual sample. This still needs more study.

Chapter 3

Bottom-up approach

3.1 Preparation

3.1.1 Motivation

A crucial observation in this thesis is that the contents in the leaf nodes of the XML documents constitute the main informative part in the document. Even though documents from different sources may follow different tag structures, and have different tag names, they are likely to have some important contents in common in the leaf nodes if they belong to the same classes. Examples are shown in Figure 3.1 and Figure 3.2, which are parts of two real XML files from the data set from [14] with the class label "automobile". These documents are from different web sites, and do not share the same DTD schema.

From Figure 3.1, we can see that most of the information in the document is in the leaf nodes. E.g. "Isuzu Ascender", "4 Speed Automatic OD", "Automatic Transmission"... etc. Actually, they are informative enough to describe certain features inde-

```
- <automobile>
    <manufacturer>Isuzu</manufacturer>
    <model>Isuzu Ascender</model>
    <year>2005</year>
  - <engine>
      <type>5.3L 8 Cylinder 300 hp Gas</type>
    </engine>
    <transmission>4 Speed Automatic OD</transmission>
  - <feature>
      <safety>ABS Brakes / Driver-Passenger airbags</safety>
      <drive>RWD</drive>
      <seat>7</seat>
      <mpg>City: 15 Highway: 20</mpg>
      <pro>Alarm</pro>
      <pro>Automatic Transmission</pro>
      <pro> Base List Price Below Average*</pro>
      <pro>CD Changer</pro>
      <pro>CD Player</pro>
```

Figure 3.1: XML document Example 1

pendently. Here, even if there were no node labels included, this leaf node information would still be enough to do the classification as long as the boundaries that separate different classes are reasonably well defined. On the other hand, some leaf nodes do need their parental node labels to be informative, such as "$< seat > 7 < /seat >$", "$< mpg > City : 15\ Highway : 20 < /mpg >$". Without the node label "$< seat >$", the word "7" itself is meaningless. It is also hard to understand what the "City: 15 Highway: 20" means when the node label "$< mpg >$" is not included. We can see here, the tag of a node provides some extra scope information for their children. E.g.: "Isuzu" is the leaf child of the node with tag "$< manufacturer >$", which limits the scope of the meaning of "Isuzu" by stating that "Isuzu" is a "manufacturer". Furthermore, the parent node of "$< manufacturer >$" is "automobile", which provides the further scope limitation that, more specifically, it is an "automobile manufacturer".

So, generally speaking, the tag of the parent node limits the semantic scope of their children, whether the child is the structural node or textural leaf node.

```
- <car>
    <generator>Oxygen XML Editor</generator>
    <database>MySQL</database>
    <id>100012</id>
    <model>Jaguar - XJ8 3.2</model>
    <manufacturer web="www.jaguar.com">Jaguar</manufacturer>
    <year>2004</year>
    <!-- features -->
  + <feature></feature>
  + <feature></feature>
    <feature>Front fog lamps Twin trip computer </feature>
    <!-- interiors -->
  + <interior></interior>
    <interior>Fixed contoured bench rear seats</interior>
    <interior>Figured walnut veneer</interior>
    <interior>Leather steering wheel</interior>
  Omitted....
    <safety>Ultrasonic intrusion sensing security system</safety>
    <safety>engine immobiliser</safety>
    <safety>Radio frequency remote control central locking </safety>
  - <power>
      <displacement>n/a cc</displacement>
      <engine>32-valve, quad cam 3.2 litre</engine>
      <installation>n/a</installation>
      <acceleration>8.5</acceleration>
      <max-power>n/a</max-power>
      <torque>n/a</torque>
      <max-speed>225</max-speed>
    </power>
  - <transmission>
      <type>5 speed electronic</type>
    </transmission>
```

Figure 3.2: XML document Example 2

In Figure 3.2, the main informative part is still in the content of the leaf nodes, such as "Fixed contoured bench rear seats", "Ultrasonic intrusion sensing security system", etc. There is also the information about the automobile manufacturer included in Example 2, but, this time, the tag of the parent node of the node $< manufacturer >$ is "car", which is literally different from the "automobile" in Example 1. Note that both

documents are from the class of automobile but have very different structures, even for similar information. For example, both contain information about "transmission", but, in Example 1, there is no "< *type* >" node under "< *transmission* >" as in Example 2. However, in the leaf nodes under "< *transmission* >" in both examples, an important word "speed" appears. As in reality, the transmission system of a vehicle is closely related to the speed adjustment. So, it is very likely that the word "speed" appears under the node that is related to the transmission system of a vehicle. On the other side, if a word "speed" shows up in the node "< *transmission* >", it is likely that this document is related to something about a vehicle.

In the later sections, "tag" or "node label" will be used interchangeably, both of which mean the label of a node.

From the above discussion, we can see that an XML document can be viewed as a text document with some supplemental element tags included. The major information lies in the leaf nodes, i.e. the textual content, and the node tags are used to define the scopes of the child element tags or textual contents beneath them. Based on these observations as well as the assumptions stated previously, we propose the XML classification method in this thesis, and use the experimental results to show the merit of our method.

3.1.2 XML representation

An XML document is usually represented by a tree (such as [8][9][27][29]) that can be denoted as N, E, R, where N is a set of label nodes, and E is a set of edges from parent nodes to child nodes. R is the root of the tree. The XML document in Figure

3.1 is represented by the following tree:

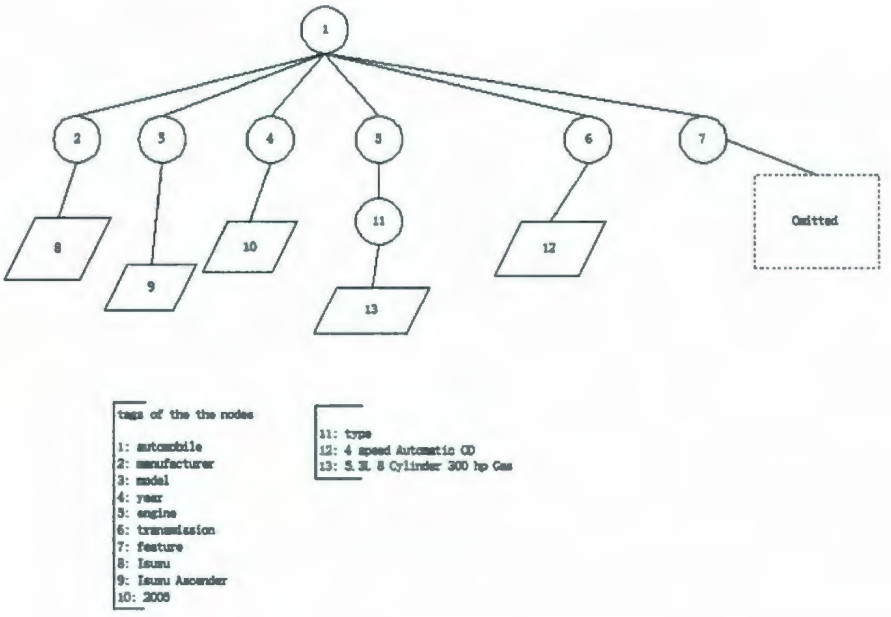

Figure 3.3: XML representation example

In Figure 3.3, the structural nodes are represented as round nodes and the leaf nodes as quadrangle nodes. The edges in the tree can be interpreted as the verb "contain". E.g.: in XML document in Figure 3.1, "< *automobile* >" contains "< *manufacturer* >", and "< *manufacturer* >" contains leaf node with the content "Isuzu", which are represented by the edges from node 1 to node 2, and from node 2 to node 8, respectively.

Additionally, some preparation is presented below.

If node A is an ancestor of node B in an XML document, then the path from node A to node B can be defined as the unique tag sequence when moving from node A to node B. When node A is the root node in this XML file, the path is called the root

path to node B. If the node B is a leaf node, the root path is called leaf node B's full path. An XML file can be represented as a collection of full paths. Each path is a unique sequence from the root node to a leaf node. The attributes of a node can also be represented as the child nodes of this node, and the values of the attributes can be the child nodes of the respective attributes themselves.

We introduce some assumptions here:

Assumption 1:

The entire data set to be classified is in XML format. The documents in it are from different sources. XML documents from different sources may not share a common DTD file, but all the schemas of the XML documents are similar if they can be classified into the same category. Here we adopt the idea of "similar DTD schema" from the homogeneity-hypothesis in [8].

In the real world, it is unlikely that all the data sources share the same DTD schema. However, if they describe the information from the same category, some popular information is likely to be used to name and structure the tags, even though they are from different data sources. For example, in XML documents related to the automobile, it is likely that "transmission", "door", "body", etc., are used to name tags, and "door" is the descendant of "body".

Assumption 2:

In each class, there are some key terms that are used exclusively to identify that class.

Classes differ from each other based on the contents included. Different classes pertain to different themes, and these themes must be distinguishable at the conceptual level. This is possible only if there exists a group of terms that collectively

characterize each class and not any other class. For example, it is likely that the terms "speed", "cylinder", "wheel", "engine" are all found in the documents in the class related with automobile but not education, environment, politics, etc.

3.2 Collecting related information from XML documents

3.2.1 Key terms

Conceptually, key terms are the terms that are highly discriminative between the containing class and other classes. Assumption 2 implies that, in each class, there exist some key terms with distinct within-class frequencies of occurrences from other classes. More specifically, the key terms for a class characterize that class. Due to this, it is important to develop strategies to search for the key terms for each class.

3.2.2 Reprocessing documents

Every leaf node in XML documents can be viewed as a group of terms contained in one leaf node, and a measuring method will be used to find the key terms within the leaf nodes. We adopt a variant version of MI (Mutual Information [2])measure here. Actually, before the beginning of the key term searching, the stop words in the leaf nodes will be filtered out. Terms in the leaf nodes are further separated by some certain characters, such as blanks and punctuations. Then the stemming method widely adopted in the text mining is also employed to transfer the different forms of a word into a normal form. In our experiment, Porter Stemming Algorithm is adopted

[3].

We discuss how to search for the key terms in a class, why the key terms can be found and what the problems of the traditional MI measure are in the following sections.

3.2.3 Term ranking and bottom-up processing

In our scheme, we calculate an MI score to evaluate the corelationship between each term and each class. In each class, different terms are sorted in descending order based on their MI scores. First m terms having sufficient occurrence in that ordered sequence in each class will be picked as key terms to that class. Then these key terms will be cooperated with the structural information to classify the document into a predefined category.

This method can be called a bottom-up method compared to the ones that usually begin from the root node of the XML document. We discuss the scheme in detail below.

3.2.4 Mutual Information (MI)

"In probability theory and information theory, the mutual information, or transinformation, of two random variables is a quantity that measures the mutual dependence of the two variables" [2]. MI can be used to measure the relevance between a term and a class. Higher MI implies higher relevance.

3.2.4.1 A general definition of MI

According to [2], an MI between two discrete random variables X and Y is defined as:

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) log(\frac{p(x,y)}{p(x)p(y)})$$

, where p(x,y) is the joint probability distribution function of X and Y, and p(x),p(y) are the marginal probability distribution fucntion of X and Y, respectively.

3.2.4.2 MI for text classification

Theobald et al. [20] employ a special version of the above formula for text mining as follows:

$$MI(X_i, C_k) = \sum_{X \in \{X_i, \overline{X_i}\}} \sum_{C \in \{C_k, \overline{C_k}\}} p(X,C) log(\frac{p(X,C)}{p(X)p(C)}) \qquad (1)$$

, where $X_i \in F, C_k \in \Gamma$, F is the set of all candidate features and Γ is the set of all the class labels in the training set. Variable X is defined as: X=X_i if a random object contains X_i, and X=$\overline{X_i}$ otherwise. Likewise, C is defined as C = C_k if a random object is labeled with C_k, and C = $\overline{C_k}$ otherwise. Thus, p(X), p(C) and p(X,C) are just traditional probability distribution functions. Intuitively, this formula measures the information that X and C share. It measures how much one of them is known if another one is given in advance. E.g., if feature X is independent to class C (meaning the presence or absence of X in an object does not alter the likelihood of the object being labeled with C), then p(X,C) = p(X)p(C), thus the formula measure is 0, which means nothing can be known on X_i (or C_k) if C_k(or X_i) is given in advance.

Based on the Formula (1), as well as the Assumption 2 earlier, the relevance of a term and a class can be evaluated using the following formula:

$$MI(t_i, C_k) = \sum_{t \in \{t_i, \overline{t_i}\}} \sum_{C \in \{C_k, \overline{C_k}\}} p(t, C) log(\frac{p(t, C)}{p(t)p(C)}) \qquad (2)$$

, where t_i and C_k are the term and the class between which the relevance is to be assessed. Variable t is defined similarly as X in Formula (1) above.

3.2.4.3 Incorporating occurrence frequencies of terms into MI

Our goal in this stage is to find the key terms in each class. As mentioned before, key terms must be informative. Formula (2) might be used to achieve this purpose. But, this formula treats t as a binary variable. This may not be appropriate if, for example, all the documents in all the classes contain t, but differ from each other in terms of its occurrence frequencies. In this case, a mere presence or absence of t may not be informative enough. To overcome this weakness, we modify Formula (2) by incorporating the occurrence frequencies of terms into it. We first introduce some definitions below.

3.2.4.4 Notations

Several notations will be used throughout the entire thesis. We put them together in Table 3.1 for easy reference.

3.2.4.5 Normalized occurrence

The percentage of a term in a document is defined as the *normalized occurrence* of that term in that document, since we have to take into account the size of a document. Generally speaking, the more terms a document has, the higher the probability that any particular term will be contained in it. A normalized occurrence relieves the

Notation	Explanation
$OCC(d,t)$	number of occurrence of term t in document d
$NOC(d,t)$	normalized occrrence of term t in document d
$ANO(C,t)$	average normalized occurrence of term t in Class C
p *touches* t	term t is included in the leaf node of the root path p
$Paths(d)$	the set of root paths in document d
$Paths(C)$	the set of key paths in class C, i.e., the class model of C
$PathSim(p,q)$	path similarity between path p and path q
$Sim(d,C)$	similarity between document d and class C
Γ	the set of predefined classes
$Keys(C)$	the key term set of class C
TS	the training set

Table 3.1: Notations and explanations

	C_1				C_2				...	C_k		
	$d_{1,1}$	$d_{1,2}$	...	$d_{1,p}$	$d_{2,1}$	$d_{2,2}$	...	$d_{2,q}$	...	$d_{k,1}$	$d_{k,2}$	...
t_1	0.0002	0	...	0	0.004	0	...	...	...			
t_2	0.006	0	...		0.0003							
...	...	...	...	...	...	...	...	...	...	...	...	...
t_m	...	...	...	...	...	...	...	...	...	...	...	...

Table 3.2: Normalized occurrences of terms in all the classes in the training set

effects by such a bias. Let t_i and d_j be a term and a document, respectively, and we have $NOC(d_j, t_i) = \frac{|t_i|}{\sum_{t_k \in d_j} |t_k|}$, where $|t_i|$ is the number of times t_i occurs, and $\sum_{t_k \in d_j} |t_k|$ is the total number of times all the terms occur in d_j.

Calculating the normalized occurrences can be done efficiently. We scan the training set once to obtain the occurrences of all the candidate terms in each document. Then their respective term percentages in every document can be evaluated, resulting in their normalized occurrences. Shown in Table 3.2 are the normalized occurrences for a sample training set. The rows correspond to terms and columns to documents. E.g.: There are k classes contained in the training set $(C_1, C_2, ...C_k)$ and p documents$(d_{1,1}, d_{1,2}, ..., d_{1,p})$ in C_1 and q documents in C_2 $(d_{2,1}, d_{2,2}, ..., d_{2,q})$...etc.

3.2.4.6 Discretization

Discretization converts a continuous variable into a categorical variable. It is necessary in classification for the discovery of patterns for the correlation of the features and class labelling. We use the supervised discretization method proposed in [11].

	C_1				C_2				...	C_k		
	$d_{1,1}$	$d_{1,2}$	...	$d_{1,p}$	$d_{2,1}$	$d_{2,2}$	...	$d_{2,q}$	...	$d_{k,1}$	$d_{k,2}$	...
t_1	I	I	...	I	II	I	...	...	...			
t_2	II	I	...		I							
...	...	...	...	...	...	...	...	...	...	...	...	...
t_m	...	...	...	...	...	...	...	...	...	...	...	...

Table 3.3: Discretized normalized occurrences to all the classes in the training set

It discretizes the normalized term occurrence into discretized sections. Each section indicates a discretized term occurrence interval. After the discretization, each normalized occurrence value is assigned to a unique section. For each term, if multiple normalized occurrence values are close to each other, they will likely be placed into a single section. This makes the MI calculation less sensitive to small differences on normalized occurrences. Those terms whose normalized occurrences differ by a big margin for different classes will take different discretized values and therefore can serve as candidates for the key terms for some classes. Shown in Table 3.3 is an example of discretized values for the normalized occurrences in Table 3.2, where I = [0, 0.003), II = [0.003, $+\infty$) for both t_1, t_2.(This might not always be the case in reality, because different terms are discretized separately. We just refer to this as an example of discretization.)

3.2.4.7 An application of MI

We propose a variant application of MI as shown in Formula (3) below:

$$MI(t_i, C_k) = \sum_{t' \in \Omega_{t_i}} \sum_{C \in \{C_k, \overline{C_k}\}} p(t', C) log(\frac{p(t', C)}{p(t')p(C)}) \qquad (3)$$

, where Ω_{t_i} is the set of categorized values for term t_i, $p(t', C)$ is the joint probability distribution function of t' and C; $p(t')$ and $p(C)$ are the marginal probability distribution functions of t' and C, respectively. These probability values can be estimated as follows. For a given $t' \in \Omega_{t_i}$ and $C \in \{C_k, \overline{C_k}\}$, $p(t', C) = \frac{Num_{i,t',c}}{Num}$, where $Num_{i,t',C}$ is the number of documents belonging to class C in which t_i takes discretized value t' (i.e., the normalized occurrence of t_i falls into section t'), Num is the total number of documents in all the classes; $P(t') = \frac{Num_{t',t_i}}{Num}$, where Num_{t',t_i} is the number of documents in which t_i has value of t'; $p(C)$ is the prior probability of class C in the training set.

In the following cases, we call the distribution of a term t in a particular class, i.e., the distribution of the frequencies of the term in the documents across that class, *the class distribution* of the term for that class. Consider the following cases.

1. The class distribution of t is similar across all the classes.
2. The class distribution of t is not similar for all the classes.

2.1 There exists a class for which the class distribution of t is different from any other class.

2.2 The class distribution of t for any class is similar to at least one other class.

An obvious question is how to interpret the words "similar" and "different" in the above definition. Like any similarity-based schemes in the field of machine learning, there does not exist an absolute definition for these terms. Their precise meanings

	C_1				C_2				...	C_k		
	$d_{1,1}$	$d_{1,2}$	...	$d_{1,p}$	$d_{2,1}$	...	$d_{2,q}$	...		$d_{k,1}$	$d_{k,2}$	...
t_1	0.002	0.0015	...	0.0017	0.00001	...	0.00002	...		0	0	0
t_2	0.001	0.001	...	0.03	0	...	0	0		0.002	0	0.001
t_3	0.015	0.0001	...	0.0002	0.001		0.0015	...		0	0	0
t_4	0.001	0.00001	...	0.005	0.00003	...	0.00001	...		0.0002	...	0.0001
t_m	...	...	...	...	...	...	...	...		...	...	...

Table 3.4: Examples of cases 2.1, 2.2. as well as the exception cases

depend on the context, and the ultimate goal of an application. For now, we can just understand them conceptually as a measure of the closeness between the class distributions of a term for different classes.

It is obvious that if a term t distributes similarly across all the classes, t cannot discriminate different classes, and therefore cannot be a key term for any class. For cases 2.1 and 2.2, let us discuss them in more detail.

In Table 3.4, t_1 is an example of case 2.1. Its normalized occurrence fluctuates at a higher level in C_1 than any other classes. Thus, it is likely that during the discretization process, t_1 will obtain distinct values in C_1 from the values it obtains in other classes. This will make t_1 informative in characterizing C_1, and therefore a good candidate for a key term for C_1.

Now, consider t_2. Its normalized occurrence fluctuates at a similar level for both the C_1 and C_k, and it shows almost no presence in any other classes, which is an example of case 2.2. Thus, t_2 cannot distinguish between these classes. This means

it cannot serve as a key term in any class.

However, the condition described in case 2.1 may not always be adequate for a term to be a key term. Still, consider t_1 in Table 3.4. We observe that t_1's class distribution in C_2 is also different from any other class. Can t_1 also serve as a key term in C_2? Intuition tells us that it should not, since its occurrence frequency is very low in C_2. It is easy to be understood that characterization must be supported by adequate presence. In our context, we request 'adequate presence' for a term in a class, as its overall volume in that class being larger than its overall volumes in any other class. That is, t_i is a key term in class C_k only if $ANO(C_k, t_i) > ANO(C_j, t_i), \forall j \neq k$, where $ANO(C, t)$ is the average normalized occurrences of t in class C. We will give the detail of this criterion later.

In an ideal situation, we would like a key term to have more or less consistent occurrences across all the documents in a class for which it is a key term. In practice, however, this may not always be the case. Consider t_3 in Table 3.4, for example. Its class distribution for C_1 and that for C_2 both meet case 2.1. Furthermore, we note that its occurrences are consistently higher in all the documents in C_2 than they are in most documents in C_1. Thus, intuitively, it is more appropriate for t_3 to be a key term for C_2 than for C_1. Notice that, however, its occurrence is very high in a single document $d_{1,1}$ in C_1. This 'over-expressed' outlier may render t_3 having a higher average occurrence in C_1 than in C_2 and therefore makes it to be mis-selected as a key term for C_1 rather than for C_2. Term t_4 describes a similar scenario, where by intuition it should have been the key term for C_1. However, because it is substantially 'under-expressed' in a single document $d_{1,2}$, it will not be selected as such. A general solution to cope with this problem is to run an outlier detection algorithm in each

class before starting the key term search. However, this will complicate our scheme. Therefore, we rather leave it as is, with the belief that extreme outliers in the context of XML document classification is rare and will not dramatically affect the overall accuracy of our scheme.

3.2.5 Substantial condition

The MI score discussed above can be used to measure whether a term is informative to a class, but it is not enough to determine the key term only by the MI score. For example, term t presents very low occurrence in class C_i, but it appears a lot in other classes $C_j, C_k, ..., j, k \neq i$. In this case, t is informative in C_i, but can we select t as a key term in C_i? The answer is definitely 'no'. It is clear that a key term needs not only to be informative to a class but enough occurrence in that class as well. For that reason, the *substantial condition* is introduced here as an extra requirement for the key terms selection in a class. Every key term in a class must satisfy this condition.

Definition [substantial condition]: Let t be an arbitrary term in class C_i, and let Γ be the predefined class set. We say that t satisfies the substantial condition with respect to C_i, if

$$\forall C_j \in \Gamma, C_j \neq C_i, [ANO(C_i, t) > ANO(C_j, t)]$$

, where $ANO(C_i, t)$ denotes the *average normalized occurrence* of term t in class C_i. $ANO(C_i, t)$ can be estimated as $\frac{\sum_{d_j \in C_i} NOC(d_j, t)}{n}$, and $NOC(d_j, t)$ stands for the Normalized Occurrence of term t in document d_j, and n is the number of documents in C_i, $NOC(d_j, t) = \frac{|t|}{\sum_{t_p \in d_j} |t_p|}$. That is, the *average normalized occurrence* of term t in class C_i is greater than any other class C_j, where $C_i,\ C_j \in \Gamma,\ j \neq i$.

3.2.6 Determining the key terms

A key term for a particular class is supposed to be informative to that class, and at the same time, it should present enough occurrence in that class, too. We use the MI value mentioned above to measure the relativity between the term and the class. Moreover, the substantial condition can guarantee the enough occurrence of the term in that class. Based on this, we make the rule for determing the key terms in a class. It has two steps: firstly, in every class, we sort all the terms based on their MI scores in descending order; secondly, we select the top m terms from the sorted list that satisfy the substantial condition. These m terms then serve as the key term set for that class.

3.2.7 Key path

In this phase, the key terms in all the classes have been determined already. We shall now utilize them with the structural information included in the tags and their dependencies. In the following, the term "path" implicitly implies that it starts from the root and ends at a leaf.

Let d be an XML document, and $d \in C$ where C is a class. Let p be a path in d. If p ends at a leaf node that contains at least one key term for C, then we call p a *key path* in C. Note that, according to the above definition, key path is a concept relating to a class, not to any individual document. However, it is defined via the individual documents belonging to the class. Thus, for any individual class, the associated set of key paths is well-defined only if (1) the documents belonging to the class are given, and (2) the key terms for the class have been determined. In the subsequent sections,

when the phrase "key path" is used, it is implicitly implied that the associated class has been given in the training set, and the key terms have been selected by the search method. Let $Paths(C)$ be the set of all the key paths in C. We call $Paths(C)$ the *class model* of C. Note that according to the above definition, a key path may end at a leaf node containing multiple key terms. Since we don't have any prior knowledge of the relationship between the key terms and the paths, we just assume they are independent of each other.

3.3 Classifying the XML documents

In our method, class prediction is based on the concept of similarities. For any given document, we define a similarity between it and each of the classes. Then, the document will be classified into the class with which it has the highest similarity among all the classes. Formally, let d be a document to be classified, and $Paths(C)$ be the class model for C, we have:

$$Class = arcmax_{C \in \Gamma}(Sim(d, C))$$

, where Γ is the set of predefined classes; $Sim(d, C)$ is the similarity between d and class C, which is calculated based on d and class model $Paths(C)$. It can be converted into similarities between paths and it is defined as:

$$Sim(d, C) =$$

$$\sum_{t \in Keys(C)} \left(\frac{1}{AVG(t)} \times \sum_{\substack{p \in Paths(d) \\ p \ touches \ t}} max\left\{PathSim(p, q) | q \in Paths(C),\ q \ touches \ t\right\}\right) \quad (4)$$

, where $Keys(C)$ is the key term set of class C; $AVG(t)$ is the average of normalized occurrences of t across all classes, $AVG(t) = \frac{\sum_{d_t \in TS} NOC(d_t, t)}{|TS|}$, where TS is the training

set, $NOC(d_t, t)$ is the normalized occurrence of term t in document d_t as defined above. Here, a path *touches* a term if the term is contained in the leaf node on this path.

3.3.1 Path similarity calculation

We have expressed the similarity between an XML document and a class in terms of the sum of the similarities between two key paths. In the following, we describe how to evaluate the latter similarities.

Let d be an arbitrary XML document, $Paths(C)$ be a class model of C, and $Paths(d)$ be the path set of d. Let $p_1 \in Paths(d)$ and $p_2 \in Paths(C)$ be two key paths. The similarity between p_1 and p_2 is determined by the three factors, denoted as α, β and γ, each of which is expressed as a function of p_1 and p_2.

Factor α is defined as:

$$\alpha(p_1, p_2) = [\sum_{s \in S}(1 + log(w(s)))]/max(|p_1|, |p_2|)$$

, where S is the maximum subset of the nodes in p_2 such that there exists a 1-1 mapping $f : S \rightarrow nodes(p_1)$ that meets the following conditions: (1) $\forall s \in S, [s$ matches $f(s)$ in tag name and type], and (2) $\forall s_1, s_2 \in S, [s_1 \prec s_2 \Rightarrow f(s_1) \prec f(s_2)]$. The expression $w(s)$ indicates the level of s in p_2.

Factor α essentially measures the extent to which the nodes in one path can match those in the other in the same order. A node is weighted with its level, since nodes at different levels have different *specificities*: the deeper its level, the more specific it is, and therefore the larger its weight. (We adopt the convention that the root is at level 1.) The α factor is reversely proportional to the lengths of the paths, since for

the same amount of node matching, the longer the paths, the less alike they are.

Factor β is defined as:

$$\beta(p_1, p_2) = [|T|]/max(|p_1|, |p_2|)$$

, where T is the maximum subset of the nodes in p_2 such that there exists a 1-1 mapping $g : T \rightarrow nodes(p_1)$ that meets the following conditions: (1) $\forall t \in T, [t$ matches $g(t)$ in tag name and type], (2) $\forall t_1, t_2 \in T \,\&\, t_1 \neq t_2, [g(t_1) \neq g(t_2)]$.

Note that we do not require g to preserve the order of the nodes in T. (This differs from S in the definition for the α factor.) Thus, the β factor measures the number of identical nodes between p_2 and p_1, the orders of which are not necessarily consistent. This captures the idea that more identical nodes between two paths makes them more similar, even if their orders are not preserved.

Factor γ is defined as:

$$\gamma(p_1, p_2) = \log(\pi(p_2) + 1)$$

, where $\forall p \in Paths(C), \pi(p) =$ the number of key terms of C that p contains.

Unlike the α and β factors, which measures the similarity of two paths, the γ factor essentially adds a weight to the similarity. This weight measures the importance of the path in the class model. The more key terms the path contains, the more important it is. Since all the paths in the class model are the paths that contain at least one key term, we have $\forall p_2 \in Paths(C), [\gamma > 0]$.

Finally, we have

$$PathSim(p_1, p_2) = \alpha(p_1, p_2) \times \beta(p_1, p_2) \times \gamma(p_1, p_2)$$

Let us see the following example to make it more clear.

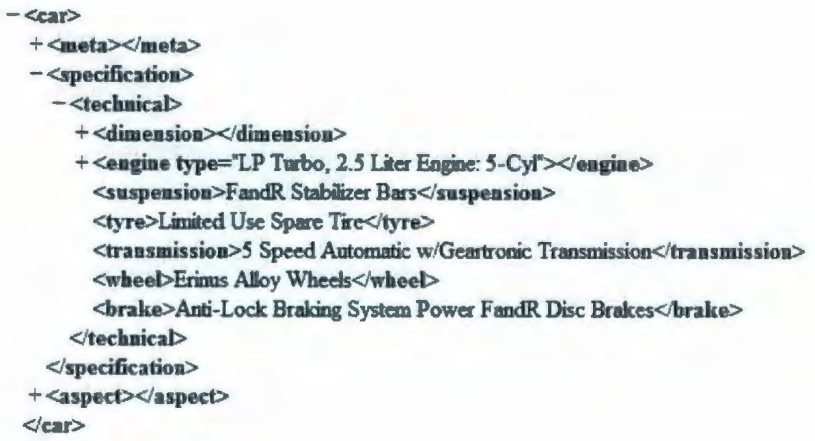

Figure 3.4: XML document Sample 1 in the testing set

For example, we have 2 XML documents in Figure 3.4 and 3.5, respectively. Suppose *Sample 1* is in the testing set, *Sample 2* is one document in the training set with class label C, and "speed" and "automatic" are two key terms in class C. To show the similarity of the paths, we take one key path that contains any of these key terms from each of the two documents. In *Sample 1*, we take the key path "$<$ *car* $><$ *specification* $><$ *technical* $><$ *transmission* $>$ [5 *Speed Automatic w/Geartronic Transmission*]", which is noted as p_1. Furthermore, we take "$<$ *car* $><$ *transmission* $><$ *specification* $><$ *type* $>$ [5 $-$ *speed EH automatic*]" from *Sample 2*, which is noted as p_2, i.e., $p_2 \in Paths(C)$.

According to the definition above, $\alpha(p_1, p_2) = \frac{(1+log(1)+1+log(2))}{5}$, since we have two identical nodes ("$<$ *car* $>$", "$<$ *transmission* $>$") in the same order in these two paths; $\beta(p_1, p_2) = \frac{3}{5}$, because without considering the order, there are three identical nodes ("$<$ *car* $>$", "$<$ *transmission* $>$", "$<$ *specification* $>$"); and $\gamma(p_1, p_2) = log(3)$, as there are two key terms contained in p_2. Since we only calculate the

```
- <car>
    ...
  - <power>
    - <specification>
        <displacement>4398 cc</displacement>
        <engine>8-cylinder light-alloy V-engine</engine>
        <installation>n/a</installation>
        <acceleration>7.0</acceleration>
        <max-power>210/286/5400</max-power>
        <torque>440/324/3600</torque>
        <max-speed>250</max-speed>
      </specification>
    </power>
  - <transmission>
    - <specification>
        <type>5-speed EH automatic</type>
      </specification>
    </transmission>
  + <fuel></fuel>
    ...
</car>
```

Figure 3.5: XML document Sample 2 in the training set

similarity between the two paths that contain the same key term, and the key paths that contain the same key term should express the similar information about the class, thus, the two paths from the same class have a higher probabilty to share more identical nodes than from different classes. We can see that, basically, for the length-normalized paths, the more the identical nodes two paths share, the "closer" they will be.

3.3.2 A justification of the similarity-based bottom-up classifier

In this section, we will justify the proposed classification scheme from a theoretical perspective. The goal is to show that it is more likely that our scheme classifies a document to the class that contains it than to any other class that does not contain it. We will use the notations listed in Table 3.1. We first make four assumptions.

Assumption I: Every term that a document contains distributes evenly among the leaf nodes in the document. Every term distributes evenly among the documents in the same class. Given two key terms from two classes, the numbers of paths that touch these two key terms in the respective class model are identical. More precisely, let t_1, t_2 be 2 key terms from C_1, C_2, respectively, $M = |\{q|q \in Paths(C_1),\ q\ touches\ t_1\}|$, $N = |\{q|q \in Paths(C_2),\ q\ touches\ t_2\}|$, then $M = N$. Given two key terms from the same class, the numbers of paths that touch these two key terms in the class model are identical too, and a document has the same amount of these 2 key terms.

We model an XML document as a tree. And we assume the term evenly distributes among the leaf nodes of a document. This assumption is a little bit strong. Since we do not have any prior knowledge of a term's distribution, we adopt a simple case, i.e., it is evenly distributed. Similarly, inside one class, the term distributes evenly in documents in the same class. For the same reason, we assume identical key term-touching path numbers as above, as well as the equality of the amount of key terms in a document.

Assumption II: The occurrence of a key term that a document contains is equal

to the number of key paths that touch this key term in this document.

Generally, for any particular key term, the more a document contains it, the more key paths related to this key term are in the document. According to *Assumption I*, a key term distributes evenly among the leaf nodes in a document. Thus, there is a proportional relationship between the numbers of key term occurrences and the key paths. For simplicity, we assume they are identical.

Assumption III: Let C_1 and C_2 be two classes of documents in the training set, d is a test document and $d \in C_1$. Let $t_1 \in Keys(C_1)$ and $t_2 \in Keys(C_2)$. Let p_1 and p_2 be key paths in C_1 and C_2 that touch t_1 and t_2, respectively. Let q_1 and q_2 be two paths that touch t_1 and t_2, respectively, in d. Then the following is true: $P(PathSim(p_1, q_1) > PathSim(p_2, q_2)) > 0.5$

Intuitively, since p_1 and q_1 both belong to the same class, while p_2 and q_2 belong to different classes, the likelihood that p_1 is more similar to q_1 than p_2 is to q_2 is higher than the likelihood for the other way around. This is consistent with our previous assumption that documents in the same class are based on more similar schemas than the documents in different classes.

Assumption IV: Let C_1 and C_2 be two classes of documents in the training set, and $d \in C_1$. Let $t_1 \in Keys(C_1)$ and $t_2 \in Keys(C_2)$. Let p_1 and p_2 be two paths that touch t_1 and t_2, respectively, in d. Let q_1 be a randomly selected key path in $Paths(C_1)$ that touches t_1 and q_2 be a randomly selected key path in $Paths(C_2)$ that touch t_2. Let $x = PathSim(p_1, q_1)$, $y = PathSim(p_2, q_2)$. Then, x follows uniform distribution in the interval [a, b], and y follows uniform distribution in the interval [c, d], and $0 < d - c = b - a$.

For a path in a document, and a path in the class model touching the same key

term, there is no prior knowledge on the relationship between them. Therefore, we just assume x can occur equally likely in an interval, i.e, [a, b]. Similarly, we assume y can occur equally likely in another interval [c, d]. In addition, without any prior knowledge about specifics about x and y, we should not be biased toward either x or y, i.e., we assume their intervals have the same width.

Since our inferences use the probabilistic framework, we first introduce four lemmas based on probability theory. These results will be used in the later context.

Lemma I: Let $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ be 2 random variable sets, where x_i and y_i are uniform random variables in the intervals [a,b], [c,d], respectively, $b - a = d - c = \delta > 0$. Assume for all $1 \leq i \leq n$, $1 \leq j \leq n$, $P(x_i > y_j) > 0.5$. Let $u = max\{X\} = max\{x_1, x_2, ..., x_n\}$ and $v = max\{Y\} = max\{y_1, y_2, ..., y_n\}$. Then: $E(u) > E(v)$.

Proof: Firstly, we claim that $a > c$. To prove the claim, we assume the contrary. Thus, $a \leq c$. $\forall x \in X$ and $y \in Y$ we have $P(x > y) = \int\int_{x>y} f(x,y)dxdy$, where $f(x,y)$ is the joint probability density function of x and y. Since x, y are independent, $f(x,y) = f(x)g(y)$, where $f(x)$ and $g(y)$ are probability density functions of x and y, respectively. Since x and y follow the uniform distribution, $f(x) = \frac{1}{b-a}$ and $g(y) = \frac{1}{d-c}$. So, $P(x > y) = \int\int_{x>y} \frac{1}{(b-a)(d-c)}dxdy = \frac{1}{(b-a)(d-c)}\int_c^b\int_y^b dxdy = \frac{1}{2}(b-c)^2\frac{1}{(b-a)(d-c)} = \frac{(b-c)^2}{2\delta^2} \leq \frac{(b-a)^2}{2\delta^2} = \frac{1}{2}$, which is a contradiction. The contradiction proves our claim.

For all i, the probability density function of x_i is, for all $x \in$[a, b], $f(x) = \frac{1}{b-a}$. Its cumulative distribution function is $F_{x_i}(x) = P(x_i \leq x) = \int_a^x \frac{1}{b-a}dx = \frac{x-a}{b-a}$. Thus, the cumulative distribution function for u is $F_u(u \leq x) = P(max(X) \leq x) = P(x_1 \leq x, ..., x_n \leq x) = \frac{(x-a)^n}{(b-a)^n}$. This implies the probability density function for u is $f_u(x) =$

$\frac{n(x-a)^{n-1}}{(b-a)^n}$. For the same reason, the probability density function for v is $f_v(y) = \frac{n(y-c)^{n-1}}{(d-c)^n}$.

So, $E(u) = \int_a^b x f_u(x)dx = \int_a^b x\frac{n(x-a)^{n-1}}{(b-a)^n}dx = \frac{n}{n+1}(b-a)+a$. Similarly, $E(v) = \frac{n}{n+1}(d-c)+c$. Since $a > c$, $E(u) > E(v)$. Thus, this lemma has been proven.

Lemma II: Let $X = \{x_1, x_2, ..., x_j\}$ and $Y = \{y_1, y_2, ..., y_k\}$ be 2 random variable sets, where for all $1 \leq i \leq j$, x_i follow the same distribution, and for all $1 \leq i \leq k$, y_i follow the same distribution, and $E(x_p) > E(y_q)$ for all p and q. Let $u = avg(X) = \frac{\sum_{i=1}^{j} x_i}{j}$, $v = avg(Y) = \frac{\sum_{i=1}^{k} y_i}{k}$, then: $E(u) > E(v)$.

Proof: $E(u) = E(\frac{\sum_{i=1}^{j} x_i}{j}) = E(x_p)$, and $E(v) = E(\frac{\sum_{i=1}^{k} y_i}{k}) = E(y_q)$, so $E(u) = E(x_p) > E(y_q) = E(v)$. This lemma has been proven.

Lemma III: Let $X = \{x_1, x_2, ..., x_m\}$, and $Y = \{y_1, y_2, ..., y_m\}$ be 2 sets of random variables where for all $1 \leq i \leq m$, x_i follows the same distribution, and so are y_i, and $E(x_p) > E(y_q)$, for all p and q. Then the following holds true: $P(\sum_{i=1}^{m} x_i > \sum_{i=1}^{m} y_i) > 0.5$.

Proof: Let S_x denote $\sum_{i=1}^{m} x_i$, S_y denote $\sum_{i=1}^{m} y_i$, and S denote $S_x - S_y$. Let $\bar{S_x} = \frac{S_x}{m}$, $\bar{S_y} = \frac{S_y}{m}$, and, $\bar{S} = \bar{S_x} - \bar{S_y}$. Because x_i, y_i follow the same distribution, according to *Central Limit Theorem*, both $\bar{S_x}$ and $\bar{S_y}$ follow the normal distribution[1]. Since $\bar{S}$ is a linear combination of $\bar{S_x}$ and $\bar{S_y}$, $\bar{S}$ follows normal distribution too, and $\bar{S} \sim N(\mu, \sigma^2)$ where $\mu = E(\bar{S_x} - \bar{S_y}) = E(\bar{S_x}) - E(\bar{S_y})$. According to *Central Limit Theorem* again, $E(\bar{S_x}) - E(\bar{S_y}) = E(x_p) - E(y_q) > 0$, so $\mu > 0$.

We now have $P(\sum_{i=1}^{m} x_i > \sum_{i=1}^{m} y_i) = P(S_x > S_y) = P(S_x - S_y > 0) = P(S > 0) = P(\bar{S} > 0) = P(0 < \bar{S} < \mu) + P(\bar{S} > \mu)$. Since $\mu > 0$, $P(0 < \bar{S} < \mu) > 0$. Thus $P(\sum_{i=1}^{m} x_i > \sum_{i=1}^{m} y_i) > P(\bar{S} > \mu)$. According to the property of normal

[1] m is reasonably big so that the *Central Limit Theorem* is valid

distribution, (i.e., $P(x > \mu_x) = 0.5$ when $x \sim N(\mu_x, \sigma_x^2)$), $P(\bar{S} > \mu) = 0.5$. So, $P(\sum_{i=1}^{m} x_i > \sum_{i=1}^{m} y_i) > 0.5$. This proves the lemma.

Lemma IV: Let C_1 and C_2 be two classes of documents in the training set, and $d \in C_1$. Let $t_1 \in Keys(C_1)$ and $t_2 \in Keys(C_2)$. Then the following is true: $\frac{OCC(d,t_1)}{AVG(t_1)} > \frac{OCC(d,t_2)}{AVG(t_2)}$.

Proof: Let $C_3, C_4, ..., C_n$ denote the rest of the classes in the training set.

Firstly, we claim that $ANO(C_1, t_1) > AVG(t_1)$, i.e., $\frac{ANO(C_1,t_1)}{AVG(t_1)} > 1$. Let $a_i = \sum_{d \in C_i} NOC(d, t_1)$, $b_i = |C_i|$, respectively, where $i \in [1...n]$. From the definition, $ANO(C_1, t_1) = \frac{a_1}{b_1}$, $AVG(t_1) = \frac{a_1+a_2+...+a_n}{b_1+b_2...+b_n}$. Let C_k denote any class other than C_1 in the training set, then $ANO(C_k, t_1) = \frac{a_k}{b_k}$ $(k \neq 1)$. From the *substantial condition*, $ANO(C_1, t_1) > ANO(C_k, t_1)$, then $\frac{a_1}{b_1} > \frac{a_k}{b_k}$. Thus $\frac{a_1+a_2...+a_n}{b_1+b_2...+b_n} < \frac{a_1}{b_1}$, implying $AVG(t_1) < ANO(C_1, t_1)$, i.e., $\frac{ANO(C_1,t_1)}{AVG(t_1)} > 1$. The claim follows.

Secondly, we claim that $ANO(C_1, t_2) < AVG(t_2)$, i.e., $\frac{ANO(C_1,t_2)}{AVG(t_2)} < 1$. Since t_2 is a key term from C_2, it is not a key term in any other class. Without knowing any specifics about its occurrences in these classes, we should put them in equal footing, by assuming it has the same average normalized occurrences in any class other than C_2. That is, $ANO(C_1, t_2) = ANO(C_3, t_2) = ... = ANO(C_n, t_2)$. Let $e_i = \sum_{d \in C_i} NOC(d, t_2)$, $f_i = |C_i|$, where $i \in [1...n]$. Then $ANO(C_i, t_2) = \frac{e_i}{f_i}$. From the *substantial condition*, $ANO(C_2, t_2) = \frac{e_2}{f_2} > ANO(C_k, t_2) = \frac{e_k}{f_k}$, where $k \neq 2$. Thus $\frac{e_1+e_2...+e_n}{f_1+f_2...+f_n} > \frac{e_1}{f_1}$. By definition, $AVG(t_2) = \frac{e_1+e_2...+e_n}{f_1+f_2...+f_n}$. Thus $AVG(t_2) > ANO(C_1, t_2)$, i.e., $\frac{ANO(C_1,t_2)}{AVG(t_2)} < 1$. The claim follows.

Thus, $\frac{ANO(C_1,t_1)}{AVG(t_1)} > 1 > \frac{ANO(C_1,t_2)}{AVG(t_2)}$. For the document $d \in C_1$, according to the *Assumption I*, we assume t_1, t_2 distribute evenly inside C_1, then, $\frac{NOC(d,t_1)}{AVG(t_1)} > \frac{NOC(d,t_2)}{AVG(t_2)}$. So $\frac{NOC(d,t_1)\cdot|d|}{AVG(t_1)} > \frac{NOC(d,t_2)\cdot|d|}{AVG(t_2)}$, where $|d|$ is the total number of terms in d. Thus, we

have $\frac{OCC(d,t_1)}{AVG(t_1)} > \frac{OCC(d,t_2)}{AVG(t_2)}$. This lemma is proven.

Based on the lemmas above, we have the following theorem.

Theorem I: Let C_1 and C_2 be two classes of documents in the training set and $d \in C_1$. Then the following is true: $P(Sim(d, C_1) > Sim(d, C_2)) > 0.5$.

Proof: From Formula (4), $Sim(d, C_1)$=

$$\sum_{t \in Keys(C_1)} (\frac{1}{AVG(t)} \times \sum_{\substack{p \in Paths(d) \\ p \text{ touches } t}} max\{PathSim(p,q) | q \in Paths(C_1),\ q \text{ touches } t\})$$

=

$$\sum_{t \in Keys(C_1)} (\frac{OCC(d,t)}{AVG(t)} \times \frac{\sum_{\substack{p \in Paths(d) \\ p \text{ touches } t}} max\{PathSim(p,q) | q \in Paths(C_1),\ q \text{ touches } t\}}{OCC(d,t)})$$

Let $OCC(d,t) = m_t$, then $Sim(d, C_1)$=

$$\sum_{t \in Keys(C_1)} (\frac{m_t}{AVG(t)} \times \frac{\sum_{\substack{p \in Paths(d) \\ p \text{ touches } t}} max\{PathSim(p,q) | q \in Paths(C_1),\ q \text{ touches } t\}}{m_t})$$

$Sim(d, C_2)$=

$$\sum_{t \in Keys(C_2)} (\frac{m_t}{AVG(t)} \times \frac{\sum_{\substack{p \in Paths(d) \\ p \text{ touches } t}} max\{PathSim(p,q) | q \in Paths(C_2),\ q \text{ touches } t\}}{m_t})$$

According to *Assumption II*, we use the occurrence of a key term to replace the number of the key path that touches this key term in the test document. And if we number the k key paths in d that touch a certain key term t from P_t^1 to P_t^k, then $Sim(d, C_1)$=

$$\sum_{t \in Keys(C_1)} (\frac{m_t}{AVG(t)} \times \frac{\sum_{k=1}^{m_t} max\{PathSim(P_t^k, q) | q \in Paths(C_1),\ q \text{ touches } t\}}{m_t}) \quad (5)$$

And $Sim(d, C_2)$=

$$\sum_{t \in Keys(C_2)} (\frac{m_t}{AVG(t)} \times \frac{\sum_{k=1}^{m_t} max\{PathSim(P_t^k, q) | q \in Paths(C_2),\ q \text{ touches } t\}}{m_t}) \quad (6)$$

Let t_1, t_2 denote 2 randomly selected key terms from $Keys(C_1)$, $Keys(C_2)$, respectively, P_t is a key path that touches t in d. Let

$f = max\{PathSim(P_{t_1}, q) | q \in Paths(C_1),\ q\ touches\ t_1\}$,

$g = max\{PathSim(P_{t_2}, q) | q \in Paths(C_2),\ q\ touches\ t_2\}$. According to *Assumption IV*, $PathSim(P_{t_1}, q) \sim U(a, b)$, and $PathSim(P_{t_2}, q) \sim U(c, d)$, $b - a = d - c$. From *Assumption III*, $P(PathSim(P_{t_1}, q) > PathSim(P_{t_2}, q)) > 0.5$. Also, from *Assumption I*, the numbers of paths that touch t_1, t_2 in $Path(C_1)$, $Path(C_2)$, respectively, are identical. Thus, the prerequisite for *Lemma I* is satisfied. Then, by *Lemma I*, $E(f) > E(g)$.

Let $r = \frac{\sum_{k=1}^{m_{t_1}} max\left\{PathSim(P_{t_1}^k, q) | q \in Paths(C_1),\ q\ touches\ t_1\right\}}{m_{t_1}}$ (7),

$s = \frac{\sum_{k=1}^{m_{t_2}} max\left\{PathSim(P_{t_2}^k, q) | q \in Paths(C_2),\ q\ touches\ t_2\right\}}{m_{t_2}}$ (8). Then, by *Lemma II*, and $E(f) > E(g)$ above, we have $E(r) > E(s)$.

According to *Assumption IV*, for all $t_1 \in Keys(C_1)$, $k \in \{1, ..., m_{t_1}\}$, and $q \in Paths(C_1)$, we have $PathSim(P_{t_1}^k, q) \sim U[a, b]$. In addition, by *Assumption I*, $|\left\{PathSim(P_{t_1}^k, q) | q \in Paths(C_1),\ q\ touches\ t_1\right\}| = n$ where n is a constant w.r.t. t_1. Let $f(x)$ be the probability density funciton for $A_{t_1} = max\left\{PathSim(P_{t_1}^k, q) | q \in Paths(C_1),\ q\ touches\ t_1\right\}$. Then from the proof of *Lemma I*, we have $f(x) = \frac{n(x-a)^{n-1}}{(b-a)^n}$. Thus, for all $t_1 \in Keys(C_1)$, $k \in \{1, ..., m_{t_1}\}$, and $q \in Paths(C_1)$, A_{t_1}s have the same probability density funciton. So, they have the same characteristic function [4]. Furthermore, by *Assumption I* as well, for all $t_1 \in Keys(C_1)$, $OCC(d, t_1) = m_{t_1} = l$ where l is a constant w.r.t t_1 and the test document d. Let the characteristic function of A_{t_1} be $\varphi_{A_{t_1}}(t)$, then the characteristic function of $\bar{A_{t_1}} = r = \frac{\sum_{k=1}^{m_{t_1}} max\left\{PathSim(P_{t_1}^k, q) | q \in Paths(C_1),\ q\ touches\ t_1\right\}}{m_{t_1}}$ in Formula (7) is $\varphi_{\bar{A_{t_1}}}(t) = (\varphi_{A_{t_1}}(t/l))^l$ [4]. So, the variables r calculated from Formula (7) follow the

same distribution. It is similar for s, too. By *Lemma III*, we have $P(R > S) > 0.5$, where

$$R = \sum_{t \in Keys(C_1)} \left(\frac{\sum_{k=1}^{m_t} max\left\{PathSim(P_t^k, q) | q \in Paths(C_1),\ q\ touches\ t\right\}}{m_t}\right),$$

$$S = \sum_{t \in Keys(C_2)} \left(\frac{\sum_{k=1}^{m_t} max\left\{PathSim(P_t^k, q) | q \in Paths(C_2),\ q\ touches\ t\right\}}{m_t}\right).$$

From *Lemma IV*, for any key term $t_1 \in Keys(C1)$ and $t_2 \in Keys(C2)$, $\frac{OCC(d,t_1)}{AVG(t_1)} > \frac{OCC(d,t_2)}{AVG(t_2)} > 0$, i.e., $\frac{m_{t_1}}{AVG(t_1)} > \frac{m_{t_2}}{AVG(t_2)} > 0$. Since each term in $Sim(d, C_1)$ is a product of the corresponding term in R and $\frac{m_t}{AVG(t)}$, and each term in $Sim(d, C_2)$ is a product of the corresponding term in S and $\frac{m_t}{AVG(t)}$, we have $R > S \Rightarrow Sim(d, C_1) > Sim(d, C_2)$. Thus, $P(Sim(d, C_1) > Sim(d, C_2)) \geq P(R > S) > 0.5$. The theorem has been proven.

Finally, the following corollary gives justification for our classification scheme.

Corollary I: Suppose there are in total n classes of documents in the training set. Let C_1 and C_2 be two classes of documents in the training set and $d \in C_1$. Let $P_1 = P(Sim(d, C_1) > Sim(d, C_2), Sim(d, C_1) > Sim(d, C_3), ..., Sim(d, C_1) > Sim(d, C_n))$, $P_2 = P(Sim(d, C_2) > Sim(d, C_1), Sim(d, C_2) > Sim(d, C_3), ..., Sim(d, C_2) > Sim(d, C_n))$. Then: $P_1 > P_2$.

Proof: Because $d \in C_1$, according to *Theorem I*, we have the following $n - 1$ formulas:

$$P(Sim(d, C_1) > Sim(d, C_2)) > 0.5 \quad (1)$$

$$P(Sim(d, C_1) > Sim(d, C_3)) > 0.5 \quad (2)$$

...

$$P(Sim(d, C_1) > Sim(d, C_n)) > 0.5 \quad (n-1)$$

Let C_x, C_y be two arbitrary classes in the training set and $C_x \neq C_y, C_x \neq C_1, C_y \neq C_1$. Since neither C_x nor C_y contains d, we should take an unbiased view toward their similarities with d. In probabilistic terms, that is, $P(Sim(d, C_x) > Sim(d, C_y)) = 0.5$. In addition, we suppose for any 3 classes, C_a, C_b, C_c, $C_a \neq C_b \neq C_c$, event $Sim(d, C_a) > Sim(d, C_b)$ is independent to event $Sim(d, C_a) > Sim(d, C_c)$ [2], i.e., $P(Sim(d, C_a) > Sim(d, C_b), Sim(d, C_a) > Sim(d, C_c)) = P(Sim(d, C_a) > Sim(d, C_b)) \cdot P(Sim(d, C_a) > Sim(d, C_c))$. Then: $P_1 = \prod_{\substack{C_x \in \Gamma \\ C_x \neq C_1}} P(Sim(d, C_1) > Sim(d, C_x))$, $P_2 = \prod_{\substack{C_x \in \Gamma \\ C_x \neq C_2}} P(Sim(d, C_2) > Sim(d, C_x))$.

Given any class $C_k \neq C_1, C_k \in \Gamma$, according to *Theorem I*, $P(Sim(d, C_1) > Sim(d, C_k)) > 0.5$, so, $P_1 > 0.5^{n-1}$. Also, because $P(Sim(d, C_1) > Sim(d, C_2)) > 0.5$, thus $P(Sim(d, C_2) > Sim(d, C_1)) < 0.5$. For the rest of the classes other than C_1 and C_2 in the training set, say, C_s, we have $P(Sim(d, C_2) > Sim(d, C_s)) = 0.5$, so $P_2 < 0.5^{n-1}$. Thus, $P_1 > 0.5^{n-1} > P_2$.

This proves the corollary.

Thus, according to *Corollary I*, given a random document $d \in C_1$, it is most likely that d will be classified into C_1.

[2]The assumption of independence may be a bit strong here. Nonetheless, it is not necessary to prove the corollary. We make this assumption to avoid tedious mathematical manipulations involved in multiple integrations on the joint distributions of the similarities.

Chapter 4

Experiment

4.1 Experiment setup

The goal of our experiment is to examine the effectiveness of our bottom-up approach. For this purpose, we compare the classification accuracies of our method with several existing works that represent the mainstream approaches in XML document classification. These include Yi's method [27], Zaki's method [29], Denoyer's method [8] and Theobald's method [20]. (In the original version of Denoyer's method, no detail is given of how to resolve the case when no match exists in the training set for a parent-child pair in a testing set. Strictly following the theory would assign it a zero probability, resulting in a poor performance. We choose to assign it the minimum probability of all the pairs ever estimated in the training set.) Our experiment is implemented by Java via the eclipse 3.2.1. We adopt the discretization implementation in weka 3.4.

Our comparison is based on two data sets. The first is a real life data set introduced

in [14], which is generated in the XML/XSLT version of web pages from 20 different sites belonging to 4 categories, labeled as "Automobile", "Movie", "Reference" and "Software". There are a total of 101 documents: 20 in "Automobile", 20 in "Movie", 41 in "Reference" and 20 in "Software". There is no cross-labeling. Since XSLT is not always accessible in real life, we ignore them in our experiment.

The second data set is collected from 10 different English news websites, which are ABC, BBC, CBS, CNN, FindNerws Org, Hosted AP Org, New York Times, Reuters, Washington Post, and Yahoo. To avoid overlapped item headings in the news of the same days in one website, i.e., one piece of news still remains active in the rest of the day, we do not collect the news more than once a day (actually, most of the news collected are 3 or more days later from the previous collecting). The RSS documents in XML format from those websites belong to 5 categories, which are labeled as "Business", "Entertainment", "Health", "Science and Technology" and "Sports". As in the first data set, every RSS file has only one class label. There are in total 274 RSS documents, 53 in "Business", 42 in "Entertainment", 60 in "Health", 67 in "SciTech" and 52 in "Sports".

4.2 Evaluation method

The evaluation is based on cross-validation. Each data set is divided into n subsets of equal sizes. For each data set, the evaluation consists of n rounds. In each round, $n-1$ subsets are used for training and the other one for testing. The overall accuracy for a data set is the average of the accuracies over n rounds. For each data set, we use two ways of dividing it into n subsets, random and by-websites. These will be

described in detail in the subsequence sections.

The metric used for accuracy evaluations is defined as follows:

$$AA = \frac{\sum_{i=1}^{n} \frac{CCN_i}{TNF_i}}{n}$$

, where CCN_i and TNF_i are, respectively, the correctly classified number of test files and the total number of test files in round i.

4.3 Experiment I: Cross-validations based on random selections

For the data sets 1 and 2, we use 4-fold and 5-fold cross validations, respectively. Samples are assigned to each subset at random. Generally, for each web site, both the training set and testing set will contain some documents from it. Since a web site uses the same DTD for the documents it generates, for any testing document, it is likely that there also exists a matching training document such that both of them share the same DTD. Generally, a random selection can guarantee that the training set and the testing set have approximately the same probability distributions for both the textual terms and the tag structures. This is favorable for algorithms that must learn not only the textual terms but also the structures.

4.4 Experiment II: Cross-validations based on web sites

This experiment uses n-fold cross-validation where n is the number of web sites from which our data are drawn [3]. In each round, we use the data from $n-1$ web sites as the training set, and the data from the other web site as the testing set. Since different web sites may use different tag structures, a document in the testing set may use a DTD that is not used by any document in the training set. In such a situation, it is hard for a learning algorithm to learn the DTDs of the testing sets from the training sets. This way of cross-validation can test the sensitivity of a learning algorithm to the un/availability of the information about the DTDs during the learning process.

4.5 Experimental results

4.5.1 Key terms

We search for the key terms from the leaf nodes in the training documents in every class. Listed in Table 4.1 and 4.2 are the top 10 key terms found in Data Set 1 and 2, respectively, in a round of cross-validation based on random selection. In the other rounds, most of the key terms selected are identical. The columns with the title of

[3]This is the case for Data Set 2. However, for Data Set 1, since several websites do not have enough sample documents, we just merge them with an other website when dividing the entire data set into n groups, respectively. We still guarantee that any 2 documents from the same website will be put in the same group. It implicitly suggests that the structures in the testing documents are "new" to the training set.

automobile		movie		reference		software	
Term	**MI**	**Term**	**MI**	**Term**	**MI**	**Term**	**MI**
CONDIT	0.1941	PERMISS	0.2348	WHEN	0.1410	WEB	0.1659
OVER	0.1928	COPYRIGHT	0.1939	MORE	0.1326	LICENS	0.1338
AGRE	0.1881	INFRING	0.1708	SAID	0.1297	XML	0.1298
LIST	0.1823	LAW	0.1639	HAD	0.1288	FILE	0.1183
SOLE	0.1823	MIN	0.1495	HEAD	0.1288	SUPPORT	0.1148
PRICE	0.1765	WARRANT	0.1494	HOME	0.1237	APPLIC	0.1147
INTERIOR	0.1765	COPI	0.1323	WHAT	0.1235	ORG	0.1096
ACCURACI	0.1708	KNOW	0.1270	HORSEPOW	0.1170	SOLUT	0.1096
VISITOR	0.1708	RECEIV	0.1241	WAI	0.1149	ALLOW	0.1094
CORRECT	0.1708	PARTI	0.1212	DRIVE	0.1082	ETC	0.1067

Table 4.1: First 10 Key terms found on Data Set 1 in one round

"MI" indicate mutual information values between the corresponding key term and the relevant class.

We note that some of the key terms in Data Set 1 are not readable under the corresponding classes. This is due to the fact that they result from the stemming process. For example, "CONDIT" actually comes from the word "conditioning". It has been used in many occasions where "air conditioning" is mentioned in *Automobile* class. "PERMISS" results from "permission". It stands for "permissible", "permission", etc, in *Movie* class. In many cases, these are used when copyrights of a movie are mentioned. Also, some terms seem not to be characteristic for the related class, such as "WHEN" under *Reference* class. However, from statistic point of view, they

Business		Entertainment		Health	
Term	**MI**	**Term**	**MI**	**Term**	**MI**
ECONOMI	0.1988	ENTERTAIN	0.2365	HEALTH	0.2737
INVEST	0.1884	JOLI	0.1524	DISEAS	0.2024
MARKET	0.1755	SINGER	0.1459	CANCER	0.1920
STOCK	0.1726	FILM	0.1400	PATIENT	0.1818
BUSI	0.1710	ALBUM	0.1272	DOCTOR	0.1748
OIL	0.1249	DRAMA	0.1270	TUBERCULOSI	0.1680
MORTGAG	0.1243	HILTON	0.1215	BREAST	0.1613
DOW	0.1223	MCCARTNEI	0.1185	FLU	0.1613
BANK	0.1146	ANGELINA	0.1185	FDA	0.1599
PRICE	0.1127	HOLLYWOOD	0.1185	DRUG	0.1591

SciTech		Sports	
Term	**MI**	**Term**	**MI**
TECH	0.2030	SPORT	0.2238
TECHNOLOG	0.1868	ROUND	0.2119
MICROSOFT	0.1725	COACH	0.2037
COMPUT	0.1598	BASEBAL	0.2001
INTERNET	0.1575	GOLF	0.1795
SOFTWAR	0.1374	TEAM	0.1754
LAPTOP	0.1339	CUP	0.1550
GOOGL	0.1299	CHAMPIONSHIP	0.1550
PHONE	0.1218	WIN	0.1452
PRIVACI	0.1214	YANKE	0.1446

Table 4.2: First 10 Key terms found on Data Set 2 in one round

are indeed informative in the training set. For example, the term "WHEN" appears 31 times in *Reference*, but substantially less times in the other classes: 0 time in *Automobile*, 11 times in *Movie*, and 1 time in *Software*. This means not only it is informative, but also has a sufficient support in *Reference*.

4.5.2 Accuracies of classifications

In Table 4.3 and 4.4, respectively, the AA scores for different learning algorithms on Data Set 1 and 2 are listed. The results are obtained by selecting 110 key terms for Data Set 1 and 60 key terms for Data Set 2. The number of the key terms is determined by the experiment, respectively.

We can see that, in all of the cases, our method has a better performance in terms of accuracies than all the other learning algorithms. From the tables, we can also observe that for both data sets, all the algorithms demonstrate a better performance in Experiment I than they do in Experiment II. As mentioned previously, in Experiment I, any testing document is most likely to have a matching training document that follows the same DTD, while this is not the case in Experiment II. This makes all the algorithms less effective in learning the structures of the testing documents in Experiment II. However, we also note that among all the algorithms, our algorithm is most resilient under such a circumstance. This is clearly demonstrated by the values in the right-most columns in both tables. This low sensitivity of our algorithm to the structural information to be learned can be attributed to our bottom-up approach, where the textual information plays a substantially more important role than the top-down-based approaches.

	Experiment I	Experiment II
Method	**AA(%)**	**AA(%)**
Yi's method in [27]	75.27	48.54
Zaki's method in [29]	40.58	40.58
Denoyer's method in [8] + upgrade	**100**	88
Theobald's method in [20]	86.15	44.45
This method	**100**	**89.12**

Table 4.3: AA scores of different methods on Data Set 1

	Experiment I	Experiment II
Method	**AA(%)**	**AA(%)**
Yi's method in [27]	96.02	83.85
Zaki's method in [29]	24.46	23.26
Denoyer's method in [8] + upgrade	93.81	81.49
Theobald's method in [20]	89.02	84.23
This method	**97.01**	**89.14**

Table 4.4: AA scores of different methods on Data Set 2

We also note that, in the general case, the decreases in performance from Experiment I to Experiment II for Data Set 1 are more severe than those for Data Set 2. This is because the documents in Data Set 1 are collected from some professional web sites, which may prefer different tag structures to be used in their documents to serve their own purposes, even for a single topic. On the other hand, the documents in Data Set 2 are collected from a group of news web sites, which have identical goals of disseminating news. Thus, the tag structures between them more or less exhibit some patterns. For example, many of them use the structural pattern like "$\langle title \rangle$", "$\langle description \rangle$", etc.

4.5.3 m-AA chart

In our method, the similarities are evaluated for the key paths, which are constructed based on the top m key terms in each class, where m is a parameter that must be supplied by users. Thus, it is of interest to examine how the performance of our algorithm depends on the number of key terms used. For this purpose, we conduct Experiment I for both data sets for different m values. Depicted in Figure 4.1 and 4.2 are the results for Data Set 1 and 2, respectively.

From the figures, we can observe that we must use sufficient numbers of key terms to achieve acceptable performances. For Data Set 1, once the number of key terms reaches 20, very high accuracy is attained. When 40 or more key terms are used, maximum accuracy is generated. For Data Set 2, the maximum accuracy is reached when 60 key terms are selected. Thus, both data sets indicate that too small a set of key terms is not adequate to characterize the containing class. However, they are

not consistent in the results generated by large key term sets. In Data Set 1, once the number of key terms reaches 50, the accuracy reaches the maximum and stays at that level, whereas in Data Set 2, when more than 100 key terms are used, the performance will deteriorate slowly as the number of key terms used increases. This can be explained as follows. The tag structures in Data Set 1 are quite discriminative from class to class. Even though additional but less informative terms are used, the structural information contained in the key paths built on the entire set of key terms is enough for accurate classifications. However, this is not the case in Data Set 2, where the tag structures are not as discriminative. Thus, when a lot of non-informative terms are used, the noises introduced into the key paths deteriorate the performances. At this point, we do not know how to derive an optimal key term set in the general case. In practice, users can select a good key term set by pooling, i.e., select the key terms incrementally, and evaluate the performance for each set. The set with the best performance can then be identified as the one for the purpose of actual classifications.

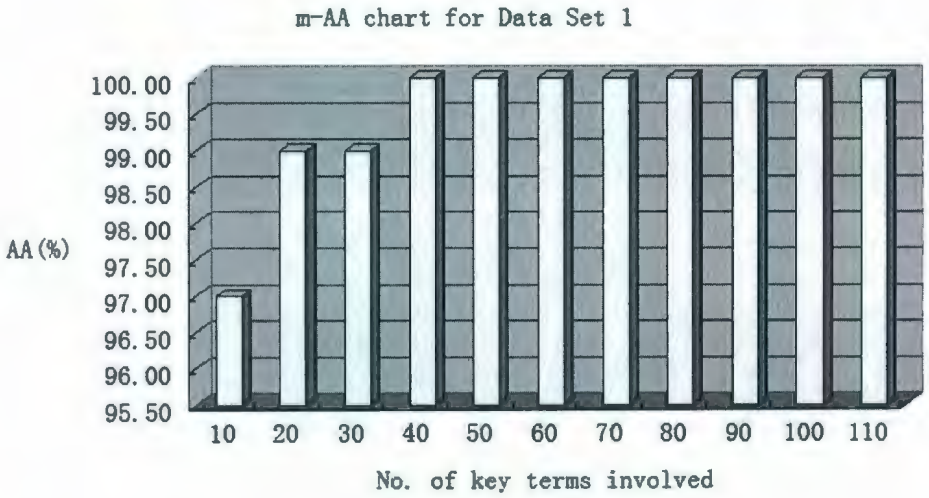

Figure 4.1: m-AA chart for Data Set 1

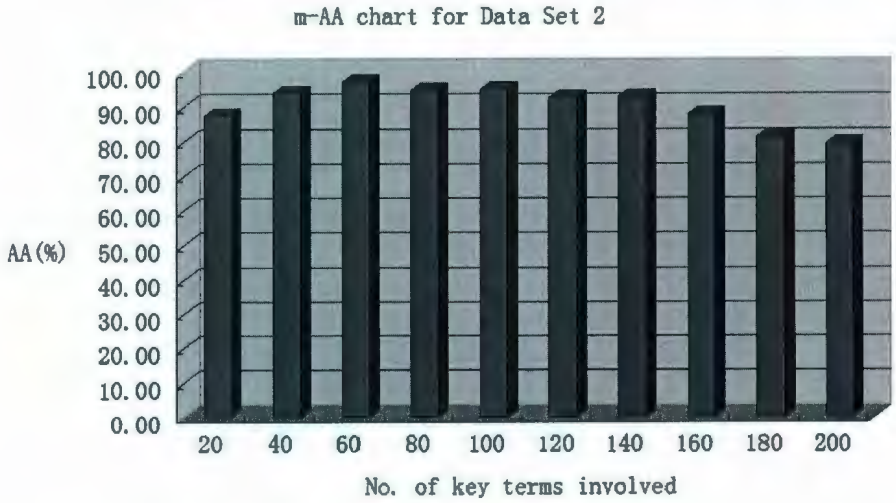

Figure 4.2: m-AA chart for Data Set 2

Chapter 5

Conclusion

We introduce a bottom-up approach for XML document classification. Our method places substantially more weight on the textual components than many existing approaches. This is based on the observation that, in many cases, the most informative components are embedded in the text. We first search for the terms that can best characterize a class. We then substantiate them with the structural information contained in the tag structures. The actual classification is based on similarity match. Our experiments show that this approach is less sensitive to the un/availability of the structural information to be learned during the learning process. This is beneficial in the cases where the documents may follow highly diverse structures.

For the XML documents whose structures present discriminative information to classes, this scheme works quite well. For the XML documents whose leaf nodes contain text information that is discriminative, this scheme performs well too. However, since different data sets have different key term sets, with respect to the classification effectiveness affected by the sizes of the key term sets, the sizes of key term sets need

to be determined case by case experimentally.

In XML document classification, structures play a very important role as we showed in the related work. We propose a new way to utilize the structure information in the tags of the XML documents. For the XML documents in the classification, there are two extreme cases, i.e., documents share the same structures or, do not have any structure in common. The former case indicates that the XML documents come from the same source, and the latter one indicates the diverse sources of the XML documents. In most of the real cases, this can be relaxed, but cannot be either of the cases. That means the real situation will be in the middle of these two cases, which is the case we assumed in our research: documents from the same category have closer structures than the ones from different categories. This thesis gives a bottom-up method to classify the XML documents in this situation. We analyze this method from both theoretical perspective and experimental results. The experimental results show it is more effective than the commonly used top-down approaches.

The future research can be in the following directions: finding the key terms more effectively and efficiently, the key terms here can be not only a single word but a phrase as well; a better measurement mechanism for the relativeness between the key terms and the categories; deciding the threshold for the document-category distance when a document can belong to multiple categories.

Bibliography

[1] http://www.w3.org/XML/.

[2] http://en.wikipedia.org/wiki/Mutual_information.

[3] http://tartarus.org/ martin/PorterStemmer/.

[4] http://en.wikipedia.org/wiki/Characteristic_function_(probability_theory).

[5] A. Bratko and B. Filipič. Exploiting structural information in semi-structured document classification. In *Proc. 13th International Electrotechnical and Computer Science Conference, ERK'2004, vol B*, pages 145–149, Portorož, Slovenia, 2004.

[6] M. Cline. Utilizing html structure and linked pages to improve learning for text categorization. In *Undergraduate Honors Thesis*, Department of Computer Science,University of Texas,Austin,TX., 1999.

[7] A. Dasgupta and et al. Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 230 – 239, 2007.

[8] L. Denoyer and P. Gallinari. A belief networks-based generative model for structured documents. an application to the xml categorization. In *Machine Learning and Data Mining in Pattern Recognition*, volume 2734, pages 277–302, 2003.

[9] L. Denoyer and P. Gallinari. Bayesian network model for semi-structured document classification. In *Information Processing and Management: an International Journal*, volume 40, pages 807 – 827, 2004.

[10] S. Eyheramendy and et al. On the naive bayes model for text categorization. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL., 2003.

[11] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conferences on Artificial Intelligence*, pages 1022–1029, 1993.

[12] D. Fragoudis and et al. Integrating feature and instance selection for text classification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 501–506, Edmonton,AB.Canada, 2002.

[13] S. Kiritchenko and S. Matwin. Email classification with co-training. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, pages 192–201, Toronto, Ontario, Canada, 2001.

[14] A. Kurt and T. Engin. Classification of xslt-generated web documents with support vector machines. In *Knowledge Discovery from XML Documents*, volume 3915, pages 33–42, 2006.

[15] P. Marteau, G. Menie, and E. Popovic. Weighted na?ve bayes model for semi-structured document categorization. In *1st International Conference on Multi-disciplinary Information Sciences and Technologies.*

[16] M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 659–661, 2002.

[17] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys*, 34:1–47, 2002.

[18] D. Shen and et al. Web-page classification through summarization. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 242–249, 2004.

[19] A. Sun, E. Lim, and W. Ng. Web classification using support vector machine. In *Proceedings of the 4th international workshop on Web information and data management*, pages 96–99, 2002.

[20] W. Theobald.M, Schenkel.R. Exploiting structure,annotation and ontological knowledge for automatic classification of xml data. In *WebDB*, San Diego,CA., 2003.

[21] W. Wolska and P. Szczepaniak. Classification of rss-formatted documents using full text similarity measures. In *LNCS*, volume 3579, pages 400–405, 2005.

[22] J. Wu and J. Tang. A bottom-up approach for xml documents classification. In *IDEAS2008*, pages 131–137, 2008.

[23] X. Wu. Xml document classification with co-training. In *ILP2007*, 2007.

[24] G. Xing. Fast approximate matching between xml documents and schemata. In *Lecture Notes in Computer Science*, volume 3841, pages 425–436, 2006.

[25] G. Xing and et al. Classifying xml documents based on structure/content similarity. In *Lecture Notes in Computer Science*, volume 4518, pages 444–457, 2007.

[26] Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, Boston, MA., 1997.

[27] J. Yi and N. Sundaresan. A classifier for semi-structured documents. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 340–344, Boston, MA., 2000.

[28] J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, Edmonton, Alberta, Canada, 2002.

[29] J. Zaki and C. Aggarwal. Xrules: an effective structural classifier for xml data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data min*, pages 316–325, Washington, D.C., 2003.