

XML Classification

by

Swathy Giri

B.E. (Computer Science and Engineering)

Madras University, Chennai, India, 2002

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Dr. Susan Gauch, Chairperson

Dr. John Gauch, Committee Member

Dr. Douglas Niehaus, Committee Member

Date Accepted _____

Acknowledgements

All thanks to GOD almighty.

I have successfully completed my Masters in computer science and this would not have been possible if it was not for my parents and my sister. They have provided me unlimited supply of encouragement and advice whenever I needed one and most of all had confidence in me to complete my higher studies here in the University of Kansas.

The path was shown by my family, but the journey was completed with the help of Dr.Susan Gauch. I am grateful to Dr.Susan Gauch for believing in my potential and for taking me under her wing, accepting to be my advisor. It was a wonderful learning experience for me during the last two years working under her as a research assistant and for my thesis. Her timely advice, on more than one occasion helped me focus better on the problem at hand and work towards an acceptable goal. Her encouraging words and support was one of the primary driving forces that kept me on the right track. I also take this opportunity to thank Dr.John Gauch and Dr. Douglas Niehaus for agreeing to be a part of my advisory committee and for taking the time to review my document and work giving valuable comments on how to make it better and to improve on the existing results

There were times when I was stopped short due to technical glitches and logical conflicts. These where the times when I was really glad that I could depend on Aravind, for lending a helping hand. I would also like to thank all my friends who stood by me and provided me with encouragement and support to successfully complete my Masters.

Abstract

As the number of XML documents on the WWW grows, there arises a need for a classification system for these XML documents that would make organization and querying more effective. This is the goal of our system that classifies XML documents based on their content. Experiments that evaluate the results of the classifier and an algorithm that can be used as the basis for classification for a previously unseen XML data set have also been outlined. Our system is shown to yield better results than a full-text classifier.

Contents

CHAPTER 1- INTRODUCTION.....	7
1.1. MOTIVATION	8
1.2 GOALS.....	9
CHAPTER 2- RELATED WORK	10
2.1. TEXT CATEGORIZATION	10
2.2. XML - BASICS AND BENEFITS.....	13
2.2.1. <i>Structure of XML documents</i>	13
2.2.2. <i>Benefits of XML</i>	14
2.3. XML CLASSIFICATION	15
CHAPTER 3- IMPLEMENTATION DETAILS.....	18
3.1. KEYCONCEPT OVERVIEW	19
3.1.1. <i>KeyConcept Categorizer</i>	19
3.2. IMPLEMENTATION	21
3.2.1. <i>Overview</i>	21
3.2.2. <i>Training phase</i>	22
3.2.3. <i>Classification phase</i>	26
CHAPTER4- EXPERIMENTAL PROCEDURE AND RESULTS.....	31
4.1. DATA SETS.....	31
4.2. EVALUATION METRIC	37
4.3. EVALUATION EXPERIMENTS WITH DS1	37
4.3.1. <i>Experiment 1- Classification with single fields</i>	38
4.3.2. <i>Experiment 2- Classification with combination of fields</i>	40
4.3.3. <i>Conclusions</i>	43
CHAPTER 5-PREDICTING VALUABLE FIELDS FOR CLASSIFICATION	44

5.1 ANALYSIS OF DS1	45
<i>5.1.1 Algorithm to predict valuable fields:</i>	48
5.2 EVALUATING DS2 WITH ALGORITHM FROM SECTION 5.1.1.....	49
5.3 EXPERIMENT 3- PREDICTING THE VALUABLE FIELDS FOR DS2.....	50
CHAPTER 6- CONCLUSIONS AND FUTURE WORK.....	52
6.1. CONCLUSIONS	52
6.2. FUTURE WORK.....	52
<i>6.2.1 Extension to multiple Schemas.....</i>	52
<i>6.2.2 Automating field selection.....</i>	53
<i>6.2.3 Further Validation</i>	53
REFERENCES.....	54

List of Figures

Figure 1: A sample XML document	13
Figure 2: Indexing in KeyConcept.....	20
Figure 3: System Architecture.....	22
Figure 4: Splitting Process.....	23
Figure 5: Indexing Process.....	25
Figure 6: Splitting Process –test documents.....	26
Figure 7: Classification Process.....	27
Figure 8: Fusion Process.....	28
Figure 9: XML schema for DS1.....	31
Figure 10: A sample XML document from DS1.....	32
Figure 11: XML schema for DS2.....	33
Figure 12: Sample XML document DS2.....	34
Figure 13: Classification accuracy for full-text vs individual fields	37

List of Tables

Table 1: Classification accuracy for full-text vs individual fields	39
Table 2: Results of experiment 2 on DS1	41
Table 3: Combinations with 2 fields providing highest accuracy.....	42
Table 4: Combinations with 3 fields providing highest accuracy.....	42
Table 5: Combinations with 4 fields providing highest accuracy.....	42
Table 6: Combinations with 5 fields providing highest accuracy.....	43
Table 7: Analysis of characteristics of fields in DS1	46
Table 8: Analysis of characteristics of fields in DS2.....	50

Chapter 1- Introduction

1.1. Motivation

The eXtensible Markup Language (XML) is becoming one of the most convenient ways to represent and transport data on the World Wide Web (WWW). Some of the features of XML that are causing such interest are: its self-describing nature; its capability to represent database schemas or object-oriented hierarchies; and the separation of formatting specifications from actual data.

Currently, the bulk of documents on the WWW are in HTML. Automatic text-classification techniques are used to help the organization and querying of these collections. This migration towards XML for data representation suggests that we might soon encounter a huge collection of XML documents on the Web. Thus, there arises the need to formulate techniques to classify these documents so that search for information on the Web can be made more efficient and effective.

Since XML documents are also text documents, a natural tendency would be to use standard information retrieval methods for classification in which each document is treated as a bag of words [1]. Alternative classification techniques consider only the structure of the XML document and are based on the premise 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 (category) [1]. These methods ignore the content of the fields altogether. Thus, they are appropriate for classifying documents that differ by

schema rather than content. In order to work, they also require that field names be meaningful and be shared across schemas.

In our approach, we have built a system that explores the use of XML-tagged data, but we classify on the basis of the tag contents (we have referred to the tag contents in this document as fields) rather than the structure. This system would work for the classification of XML documents that are marked up to the same schema or similar schemas that can be migrated to a common subset.

1.2 Goals

The goals of this project are,

- 1) To develop a system that aids in performing classification based on
 - a. Individual fields
 - b. Weighted combination of fields
- 2) Confirm our hypothesis that when classifying XML documents, fields matter and some fields matter more than others.
- 3) Arrive at a heuristic that can be used to select weight fields for classification *a priori*.
- 4) Validate the field-weighting heuristic on previously unseen collection of XML documents.

Chapter 2- Related Work

In this chapter, we review existing work on plain text classification and XML classification. Section 2.1 outlines existing text categorization algorithms. Section 2.2 describes the basic features and benefits of XML documents and Section 2.3 provides a brief description of work that has been conducted in the XML classification domain.

2.1. Text Categorization

Text Categorization is the assignment of text documents to predefined categories/classes. There are 2 phases involved in this process: *training* and *classification*. In the training phase, sets of documents belonging to each category are used to create representations of the categories. The classification phase deals with comparing these representations with a new document in order to assign the new document to one or more categories. There are a wide range of algorithms that can be used to perform text categorization, including support vector machines (SVM) [8], k-nearest neighbor classification (kNN) [4], neural networks [7], linear least squares fit mapping (LLSF) [19], naïve bayes classification (NB) [5] and the vector space method [9]. [3] compares the performance of all these classification systems and concludes that SVM and kNN significantly outperform all other classifiers and NB underperforms all other classifiers.

In *kNN classification* [4], given a test document the system finds the k nearest neighbors among the training documents. A similarity score between a document and all other documents are calculated. These scores are sorted by values and the top k matches are

found. The categories to which each of the top k matches (neighbors) belong are already known. If more than one neighbor belongs to the same category then their scores are added up to get an overall score for the category. Finally these scores for each category are ranked to obtain an estimate of the document belonging to each of the categories.

A typical **NB Classifier** [5] uses the joint probabilities of words and categories to estimate the probabilities of categories given a document. It makes an assumption that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category.

[7] describes a **neural network** approach to classification, wherein text is first transformed into a feature vector representation. This process requires an appropriate choice of features to use in feature vectors. Feature vectors that are derived from relevant texts of a particular category form the training examples for that category. Next, an automated learning algorithm learns the necessary association knowledge from the training examples to build a classifier for each category.

SVM is a machine-learning algorithm that treats learning as an optimization problem [8]. Every document used for training is represented as a vector. For each category, each training document either belongs to its positive class or negative class. The learner then attempts to find a boundary that achieves the best separation between these two classes. When a new document arrives, it is categorized by calculating its distance from the boundary.

The **linear least squares fit mapping** [19] attempts to automatically learn a multivariate

regression model from a training set of documents and their categories. The training data are represented as input/output vectors representing words/categories respectively and a linear least-square fit is solved on these training pairs to obtain a matrix of word-category regression coefficients. A document that is to be categorized utilizes these coefficients to obtain a list of weighted categories to which it belongs.

In our project, we make use of the classifier created for the KeyConcept project [9]. It uses **a vector space model** for classification. In the vector space approach, a document is treated as a vector of weighted terms. The weight of each term in a document is calculated by using the formula

$$WEIGHT_{jk} = FREQ_{jk} \times \log(n / DOCFREQ_k) \quad [10]$$

Where, $WEIGHT_{ik}$ - is the weight of term k in document i,

$FREQ_{ik}$ – is the frequency of the word k in document i,

n- is the number if documents in the collection, and

$DOCFREQ_k$ – is the number of documents in which k occurs.

This function embodies the intuitions that (i) the more often a term occurs in a document, the more it is representative of its content, and (ii) the more documents a term occurs in, the less discriminating it is [11].

Thus, for each category, a set of pre-classified documents is used during the training phase to determine words that are representative of the category. Essentially, each category is represented by a vector that is the centroid of the training documents for that category. When a new document arrives, it is assigned to one or more of the categories depending on the terms it contains. The similarity between the document vector and the category centroids are calculated using the cosine similarity measure equation and the

document is assigned to the N most similar categories.

2.2. XML - Basics and Benefits

XML is a *standard, simple, self-describing* way of encoding both text and data so that content can be processed with relatively little human intervention and exchanged across diverse hardware, operating systems, and applications [12]. XML was originally developed from the need to improve the functionality of web technologies through the use of a more flexible and adaptable means to identify information [13]. It is also known as a *meta-language*, since it holds both information (data) and description about the information (meta-data). These characteristics make XML a beneficial tool in many applications.

2.2.1. Structure of XML documents

An XML document contains text and format markup. The format markup can be either plain fields or fields containing 1 or more attributes. Figure 1 shows a sample XML document.

```
<company_info>
<location>Lawrence,KS</location>
<phone type="voice">785-345-6785</phone>
<type> Computer Hardware </type>
</company_info>
```

Figure 1- A sample XML document

An XML document should conform to the *well formedness* constraints [14]. This means that in order to be well-formed, the XML document should conform to the XML syntax rules that require them to follow proper nesting, have a unique opening and closing field that contains the whole XML document, and follow proper case for all the fields (XML is case sensitive). A *valid XML document* is one that is well formed as well as conforms to a *DTD (Document Type Definition)* or a *XML Schema*. These contain a set of rules that define what fields can appear in a XML document and also the structure of XML documents.

2.2.2. Benefits of XML

[13] outlines the benefits of XML that make it an effective solution for the design of a wide range of applications and web services.

- **Simplicity**

Information coded in XML is easy to read for humans and understand, and easy for computers to process.

- **Openness**

XML is a W3C standard, endorsed by software industry market leaders.

- **Extensibility**

There is no fixed set of fields. New fields can be created, as they are needed.

- **Self-description**

In traditional databases, data records require schemas set up by the database administrator. Because they contain meta-data in the form of fields and attributes, XML documents can be stored without such definitions XML also provides a

basis for author identification and versioning at the element level. Any XML field can possess an unlimited number of attributes such as author or version.

- **Supports multilingual documents and Unicode**

XML is appropriate for the international applications.

- **Facilitates the comparison and aggregation of data**

The tree structure of XML documents allows documents to be compared and aggregated efficiently, element-by-element.

- **Can embed multiple data types**

XML documents can contain any possible data type - from multimedia data (image, sound, video) to active components (Java applets, ActiveX).

- **Can embed existing data**

Mapping existing data structures like file systems or relational databases to XML is simple. XML supports multiple data formats and can cover all existing data structures.

2.3. XML Classification

Due to the increasing proportion of XML documents on the web, attempts are being made to design categorization algorithms for XML documents in order to make information organization and search more effective. In this section we review a few techniques that have been developed for the classification of XML documents.

[1] proposes a classification technique using *hierarchical taxonomies*. These are

essentially concept hierarchies that arrange concepts into a tree-like structure for better abstraction [16]. Each node in the tree would be *a priori* populated with a number of prototypical documents derived from user bookmarks, surf trails, and other profiling information. These documents serve as training data for the classifier. The project follows a two-step paradigm, known as ***focused crawling***, that interleaves crawling and classification. A focused crawl starts with the training documents for each of the taxonomy's categories as seeds and then traverses a small fraction of the Web linked from these training documents. The collected documents are then classified and either added to one or more categories when the classification test is positive or discarded if no string match is found.

When a new document arrives, a top-down procedure is used for classification. Starting from the root of the taxonomy, for each category considered a support vector machines classifier using a supervised learning algorithm, is invoked and it returns a yes-or-no decision and a confidence measure of the decision. When the document fits with more than 1 category, either the one with highest confidence measure is chosen or the document is classified into multiple categories. Then the classification proceeds with the children of the categories to which the document is added.

This project is implemented in a prototype focused crawler called BINGO! (for bookmark-induced gathering of information). Experiments to evaluate the benefits of the system for classifying XML documents are being carried out.

A belief networks-based generative bayesian model is used by [18] to categorize XML

documents. This technique considers both the structure and the content information contained in XML documents. The approach represents a structured document as a Directed Acyclic Graph (DAG) wherein each node of the graph represents a structural entity of the document, and each edge represents a hierarchical relation between the two entities (for example, a paragraph is included in a section, two paragraphs are on the same level of the hierarchy, etc). This representation leads to the existence of three types of information in a document: the logical structure represented by the arcs of the DAG, the label information, and the textual information. A Bayesian network model is constructed with the components for computing the structural and textual probability. A final belief network is built by combining both the components. A machine learning approach is used and the model parameters for each category are learned from a pre-classified training set of representative documents. This model is then used to predict the probabilities that a new document will belong to a particular category and the best match is chosen. This model was originally designed for documents belonging to a single DTD but has been slightly modified to classify documents from more than 1 unknown DTD's. The researchers evaluated their system using the webKB collection [17] of documents and transformed them into XML documents belonging to 6 categories and have used a naïve bayes model as a baseline system. They have obtained an increase of approximately 3% by using the belief network classifier.

[20] discusses a structural classifier for XML documents termed as ***XRules***. A set of *structural rules* are constructed in the training phase by identifying a set of representative structural patterns for each category. It is based on the premise that the presence of a particular kind of structural pattern in an XML document is related to its likelihood of

belonging to a particular category. During classification, all the rules relevant to each new document are identified and the statistics from all the matching rules are combined to predict the most likely category for each document. This classifier has been evaluated for both real (constructed from Log reports) and synthetic (constructed by a data generation program simulating website browsing behavior) data sets and has been compared to a traditional vector space classifier and an association based classifier [21]. They have shown that XRules outperforms both the classifiers; its accuracy is 2-4 % better when evaluated for the real datasets and about 20% better for synthetic datasets.

Chapter 3- Implementation Details

In this section, we review our pilot work and provide implementation details for our project. Section 3.1 provides an overview of the KeyConcept project, that we have adapted according to our requirements, and Section 3.2 explains the system architecture in detail.

3.1. KeyConcept Overview

As the number of available Web pages grows, users experience increasing difficulty finding documents relevant to their interests. One of the underlying reasons for this is that most search engines find matches based on keywords, regardless of their meanings. To provide the user with more useful information, we need a system that includes information about the conceptual frame of the queries as well as its keywords. This is the goal of KeyConcept, a search engine that retrieves documents based on a combination of keyword and conceptual matching. Documents are automatically classified to determine the concepts to which they belong. When a query is given, the concept to which it belongs can either be specified explicitly or can be determined by a user profile.

3.1.1. KeyConcept Categorizer

Figure 1 shows the indexing process in KeyConcept. It is comprised of two phases: classifier training and collection indexing. During classifier training, a fixed number of sample documents for each concept are collected and merged, and the resulting

superdocuments are preprocessed and indexed using the vector space model. This essentially represents each concept by the centroid of the training set for that concept. During collection indexing, new documents are indexed using a vector-space method to create a traditional word-based index. Then, the document is classified by comparing the

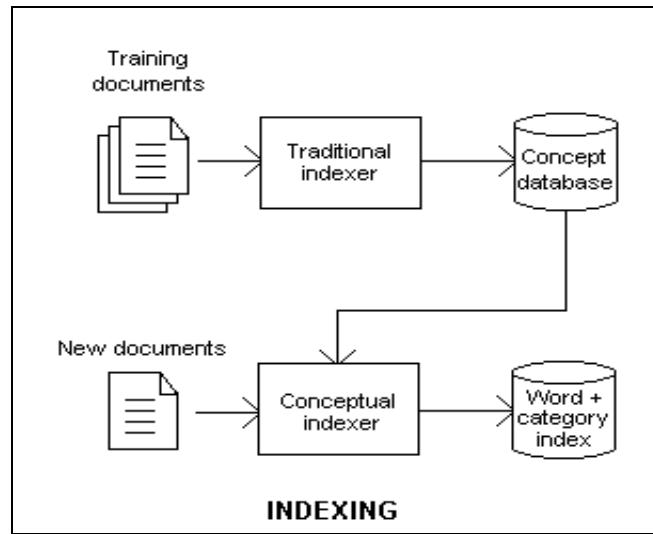


Figure 2: Indexing in KeyConcept

document vector to the centroid for each concept. The similarity values thus calculated are stored in the concept-based index.

In our project, we have extended the KeyConcept categorizer to classify XML documents using all or some fields or ignoring the XML structure altogether. These extensions were implemented as a set of wrappers around the KeyConcept classifier.

3.2. Implementation

3.2.1. Overview

The basic hypothesis of our project is that we might obtain better results when we classify XML documents paying attention to the contents of some fields more than the others. To test this hypothesis, we have designed a XML classifier that first splits the XML documents during the *training phase* into N documents, where N is the number of fields in a document, in which each of the documents contains the content of a single field from the main XML document. The next step in the training phase is to train the classifier and build indexes with the content from each of the fields separately. The indexes created in this phase are then used to classify test documents in the *classification phase*, after they go through the same splitting process as the training documents. As a baseline system for our experiments, we classified the full XML documents using the unmodified KeyConcept classifier, ignoring the document structure grouping all content together. Figure 3 shows the overall system architecture.

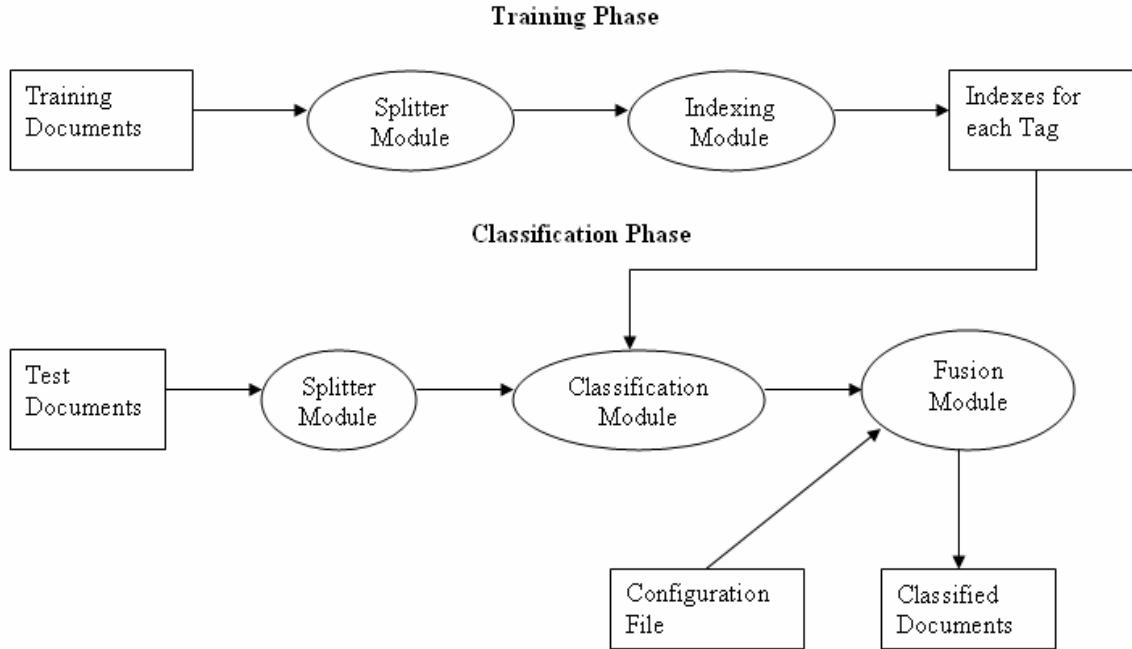


Figure 3: System Architecture

3.2.2. Training phase

Splitter Module

This module splits the fields in the document and creates a new document for every field in the main XML document. The module has been written in Perl and uses a special XML module in Perl called XML: TWIG. It takes as input the name of a directory that contains one sub-directory per category containing the XML training documents for that category. The module creates a new directory for every field per category containing training documents that contain the data for that particular field alone. These new training directories are later used by the indexer module, to create separate indexes of each for the fields.

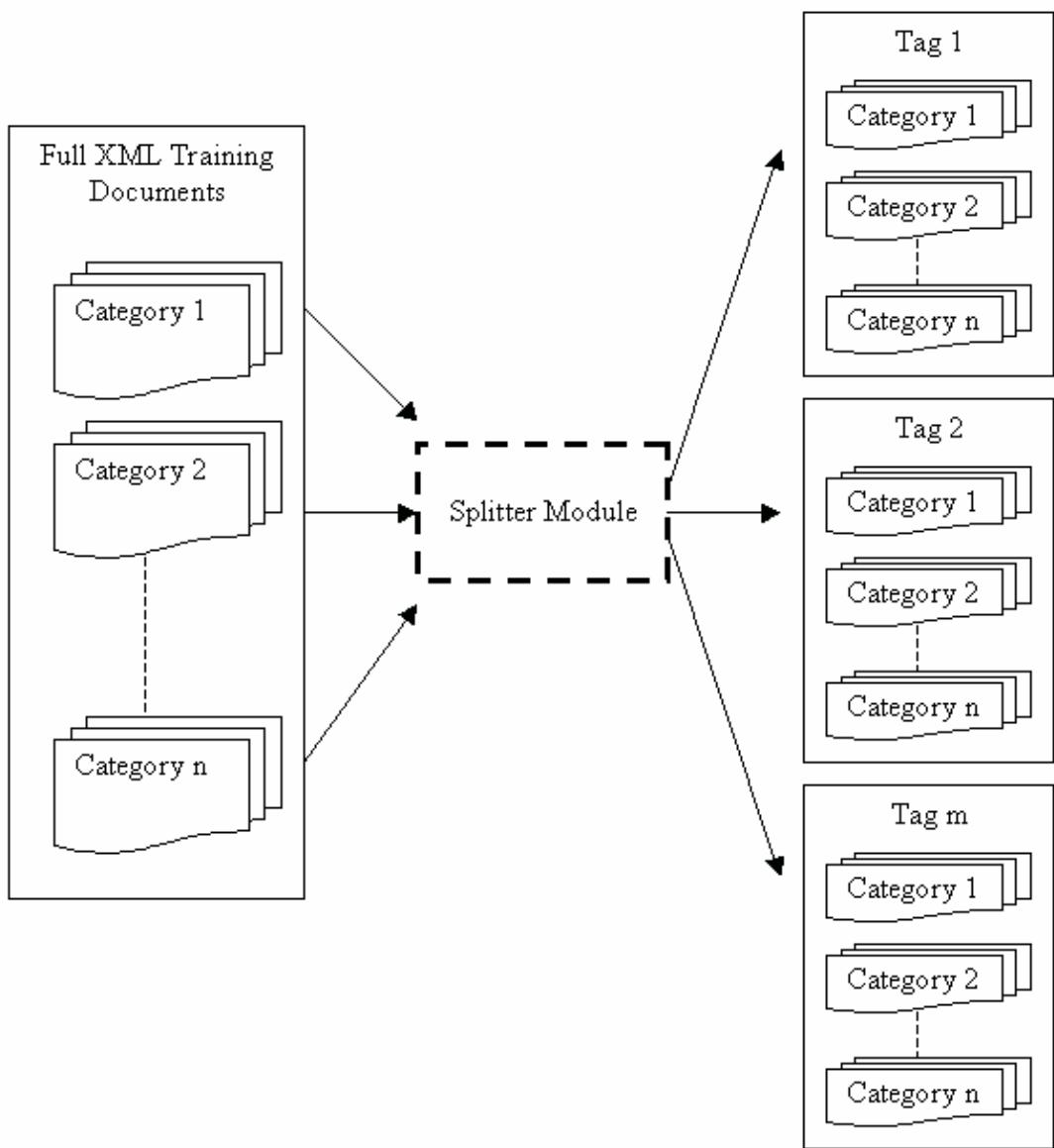


Figure 4: Splitting Process

Figure 4 represents the splitting process accomplished by the splitting module. Each XML document contains a subset of m fields; hence the resulting directories for each field may or may not have a document depending on whether or not that particular field was present in the original XML document.

Indexer Module

We have used the indexer designed for the KeyConcept project for this module. KeyConcept uses the vector space model discussed in section 2.1 for classification of documents. The training documents belonging to the same category are combined into a superdocument, which is considered representative of documents for that category. The result of the indexing process is an *inverted index* that stores words extracted from the input files and also stores statistical information about them. There are three files that combine to form an inverted index of a collection of documents. They are the *dictionary* file (dict), the *postings* file (post) and the *documents* file (docs). The dictionary file stores the word, its IDF (inverse document frequency) and a pointer to the first occurrence of the word in the postings file. The postings file stores the number of times a word has occurred in a document. The documents file stores the mapping between the document id and each word contained in the dictionary and the postings file. Finally we have normalized the results by the length of the document (stored in the documents file).

The inputs to the indexer are the output files from the splitter module. Figure 5 shows the diagrammatic representation of the inputs and outputs for the indexer module.

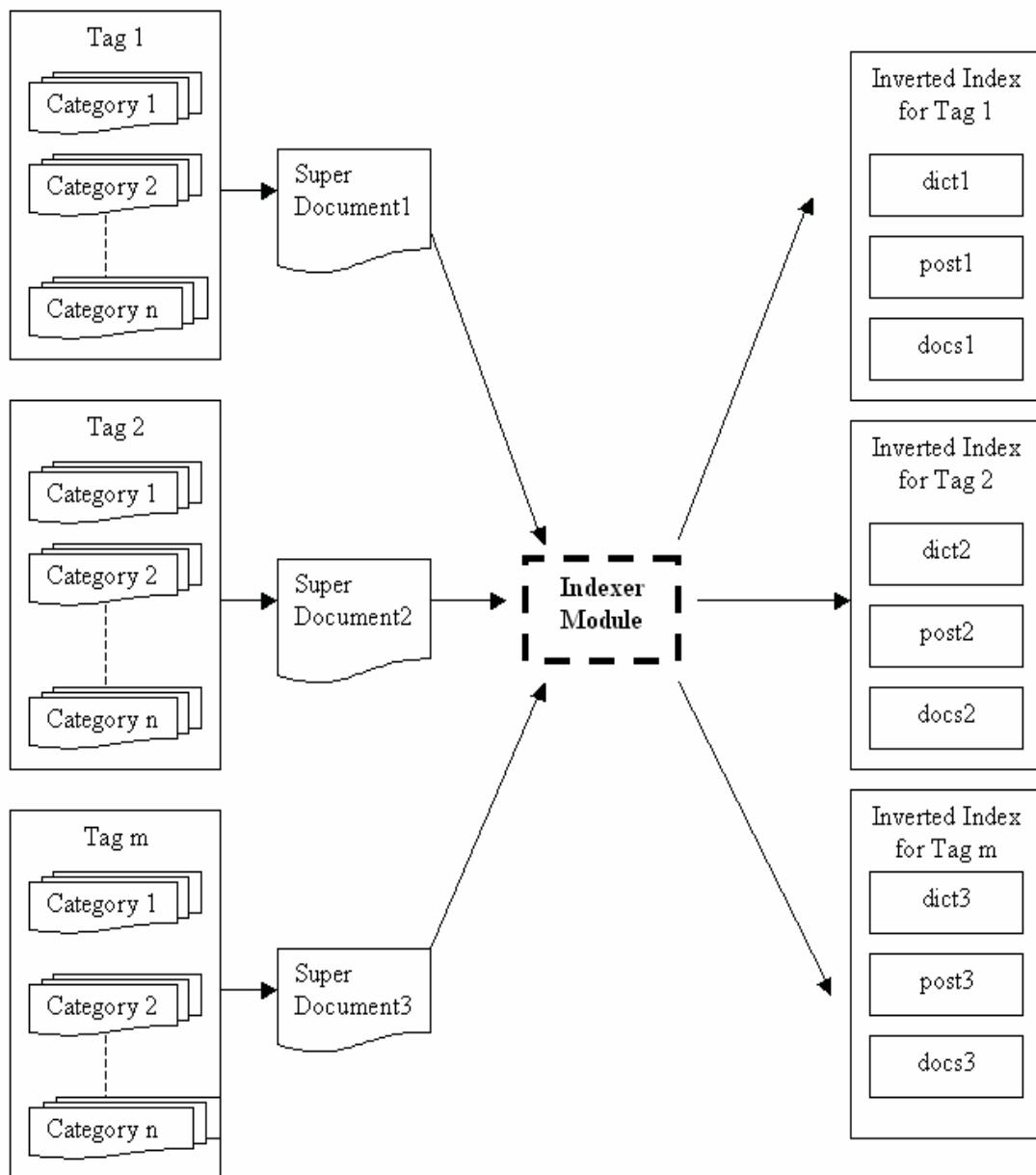


Figure 5: Indexing Process

Figure5: Indexing Process

3.2.3. Classification phase

Splitter Module

The splitter module for the classification phase is the same that is used in the training phase with the only difference being that full XML test documents are given as input to the splitter and it splits them in the same way as the training documents, creating a new file for every field in each of the test documents. The output documents from this module are used for classification.

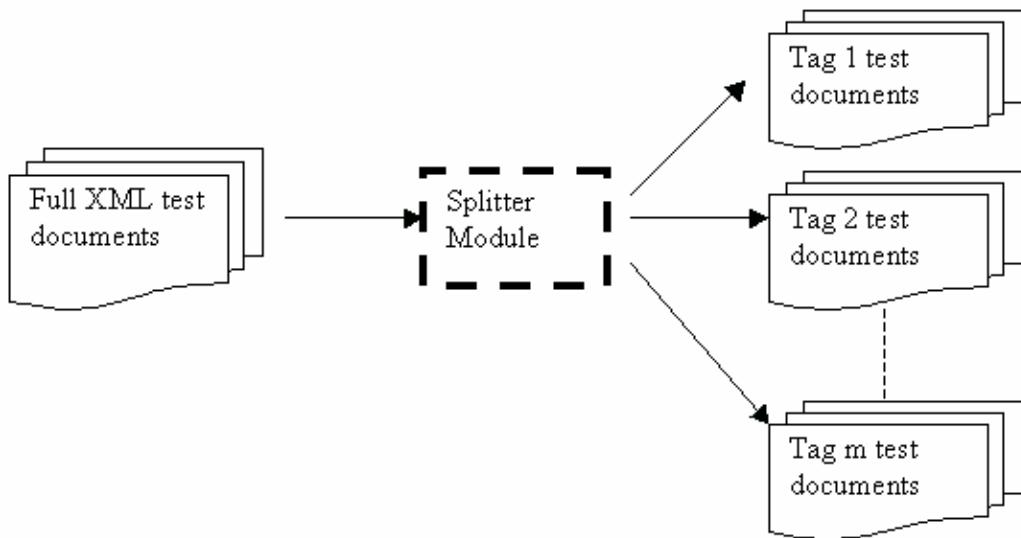


Figure 6: Splitting Process –test documents

Classification Module

This module makes use of the inverted index created on a per field basis in the training phase to classify the split test documents. This module has to be invoked for the set of test documents for each field and the path of the appropriate inverted index also has to be specified. The new documents are then compared to the inverted index and the top k concepts and their weights are determined for every test document. The output of the classifier module is a file containing the list of Category ID's and the corresponding weights per test document. This list is sorted by weight to obtain the most relevant categories. k can be specified to the module depending on how many concepts with which we would like a document to be associated.

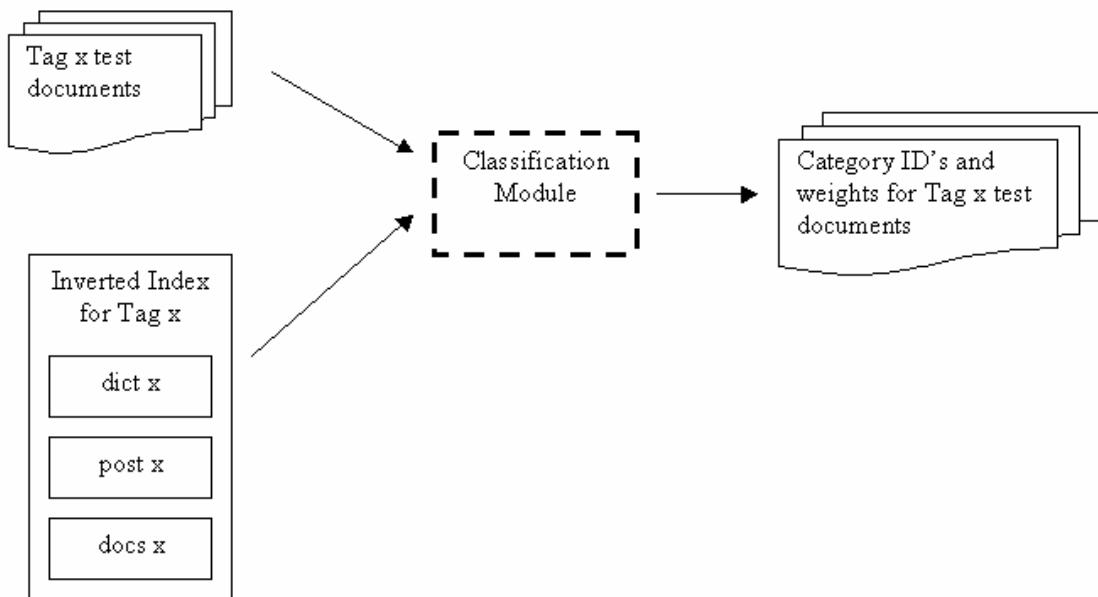


Figure 7: Classification Process

Fusion Module

The fusion module is used in our project to perform a weighted combination the results obtained from the classification module. This is done to see if combinations of the results obtained from a subset of fields yield a better result for the classification problem than classifying with respect to each of the fields in isolation. The inputs to the fusion module are the result files from the classification module, a path_configuration file that contains the list of directories that contain the results from classification with respect to each field, and a weights_configuration file that has a list of possible weights that need to be applied to the classification results with respect to each field. The output of this module is also a list of category ID's and weights for every test document sorted by weights.

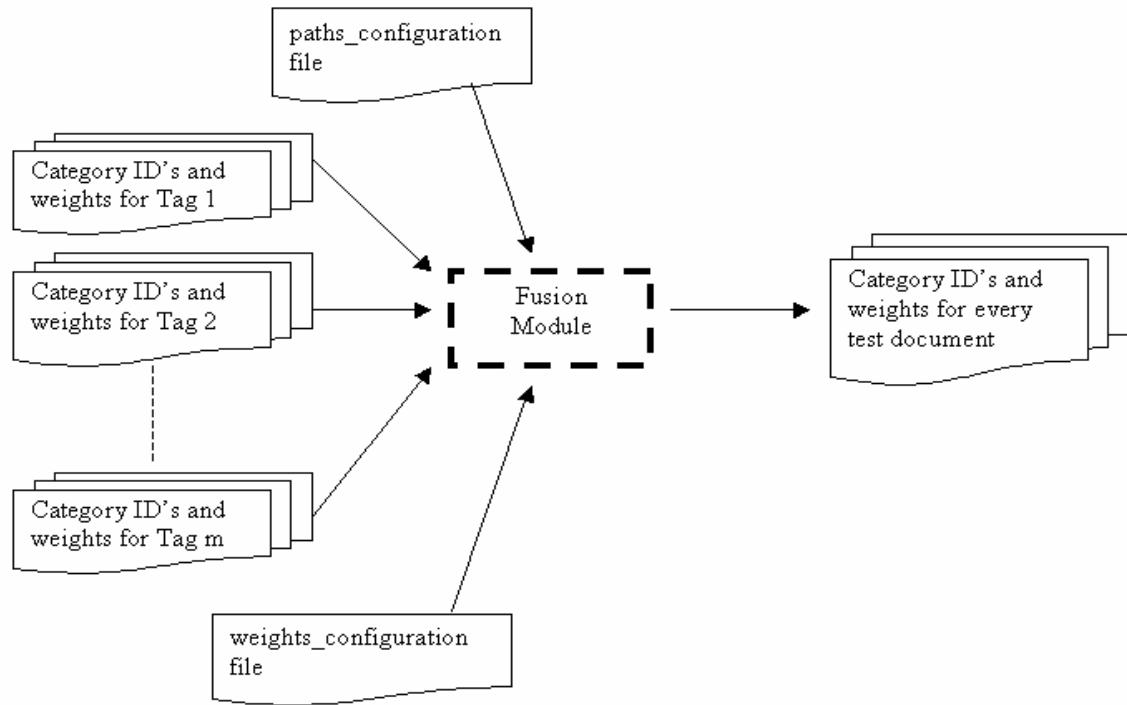


Figure 8: Fusion Process

Figure8: Fusion Process

The combined weights for each category i are obtained by performing the following calculation:

$$\text{combined_weight}_i = (W_1 * \text{field1_weight}_i) + (W_2 * \text{field2_weight}_i) + \dots + (W_m * \text{field}_m\text{weight}_i)$$

Where,

W_1, W_2, \dots, W_m are weights given to each of the m fields in the schema.

Normalization of the weights is carried out on a per document (field) basis before combining them by dividing them by the top weight of every result file from the classification module. Thus, each field has a contribution of 0...1.0.

Chapter4- Experimental Procedure and Results

Section 4.1 of this chapter deals with the data sets used in our project. The evaluation metric is explained in Section 4.2 and Section 4.3 outlines the evaluation experiments performed on data set 1.

4.1. Data Sets

We have created two data sets, each containing a set of 160 XML documents, each using a consistent but distinct XML schema. One of these sets has been used for experimentation and the other for validation purposes. The documents in each set belong to 4 different categories, with 40 documents per category. 30 documents per category have been used for training and 10 documents from each category have been used for testing.

Data set 1 (DS1) contains news articles collected from two websites, www.bbc.com and www.rediff.com. The schema for this data set is shown in Figure 9. We created the schema to capture the important aspects of the content from these websites. News, Business, Science and Health are the categories from which we downloaded between January 2003 to March 2003. These documents were then manually annotated with the tags from the schema that were appropriate for the documents. Figure 10 shows a sample document from this data set. DS1 was used in our initial experiments to determine which fields were most useful for categorization.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="rss">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="pubdate"/>
        <xs:element ref="copyright"/>
        <xs:element ref="creator"/>
        <xs:element ref="language"/>
        <xs:element maxOccurs="unbounded" ref="item"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="pubdate" type="xs:string"/>
  <xs:element name="copyright" type="xs:NCName"/>
  <xs:element name="creator" type="xs:string"/>
  <xs:element name="language" type="xs:string"/>
  <xs:element name="item">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="description"/>
        <xs:element ref="link"/>
        <xs:element ref="details"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="description" type="xs:string"/>
  <xs:element name="link" type="xs:anyURI"/>
  <xs:element name="details" type="xs:string"/>
</xs:schema>
```

Figure 9: XML schema for DS1

```

<rss>
<pubdate>3rd September 2003</pubdate>
<copyright>Rediff</copyright>
<creator>Swathy Giri</creator>
<language>Eng-United States</language>
<item>
    <title>Budget opens party battle lines</title>
    <description>Gordon Brown is set to defend Budget plans to shed 40,000 Whitehall jobs so he can provide more schools cash.</description>
    <link>http://news.bbc.co.uk/go-click/rss/0.91/public/-1/hi/uk\_politics/3521716.stm</link>
    <details>Delivering his eighth Budget, the chancellor froze a range of taxes and claimed the UK was enjoying its longest period of economic growth since the Industrial Revolution. People over 70 will get a one-off Â£100 to help cope with council tax rises, while primary schools each get Â£55,000 for school improvements and secondary schools get Â£180,000. </details>
</item>
<item>
    <title>Oil prices surge to 13 year high</title>
    <description>The price of crude oil has increased to its highest level since October 1990 as terrorism fears continue to worry the market.</description>
    <link>http://news.bbc.co.uk/go-click/rss/0.91/public/-1/hi/business/3521952.stm</link>
    <details> US crude oil prices have reached their highest level since 1990 as ongoing global terrorism fears continue to put an upward pressure on the market. With last week's events in Spain still fresh in the mind of traders, the price of New York's benchmark light sweet crude surged 70 cents to $38.18. This is the highest figure since 16 October 1990 - more than 13 years ago in the run up to the first Gulf War. The upward pressure is also being driven by concern of low US stocks. Production targets It is at this time of the year that America usually stockpiles both crude oil and petrol ahead of its main consumption season in the summer. Yet a number of recent official US government reports have said that stocks are currently at historic lows.
    </details>
</item>
</rss>

```

Figure 10: A sample XML document from DS1

Data set 2 (DS2) contains information about four different categories of companies on the WWW, i.e., Hardware, Technology, Advertising and Marketing, and Cosmetics. We created the schema for this data set, shown in Figure 11, to capture important aspects of the content. As in the case of DS1, the web pages collected about companies in these categories were manually annotated with XML tags. Figure 12 shows a sample XML

document from DS2. We used DS2 for validation of the algorithms we developed based upon experiments with DS1.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="company">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" />
        <xs:element ref="url" />
        <xs:element ref="headquarters" />
        <xs:element ref="branch" />
        <xs:element ref="products" />
        <xs:element ref="services" />
        <xs:element ref="datevisited" />
        <xs:element ref="creator" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string" />
  <xs:element name="url" type="xs:anyURI" />
  <xs:element name="headquarters">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="location" />
        <xs:element ref="phone" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="branch">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="location" />
        <xs:element ref="phone" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="products">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="product" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="product" type="xs:string" />
  <xs:element name="services">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="service" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="service" type="xs:string" />
  <xs:element name="datevisited" type="xs:string" />
  <xs:element name="creator" type="xs:string" />
  <xs:element name="location" type="xs:string" />
  <xs:element name="phone" type="xs:string" />
</xs:schema>

```

Figure 11: XML schema for DS2

```

<company>
    <name>Allegis Corp. </name>
    <url>http://www.allegiscorp.com/jsp/displayTypes.jsp?cat=HVAC--Hardware</url>
    <headquarters>
        <location>Allegis Corporation,P.O. Box 49007,Minneapolis, MN 55449
    </location>
        <phone>1-866-378-7550</phone>
    </headquarters>
    <products>
        <product>Ball Bearing Drawer Slides </product>
        <product>Casters </product>
        <product>Door Assists </product>
        <product>Folding Steps </product>
        <product>Handle </product>
        <product>Heavy Duty Drawer Slides </product>
        <product>Hooks </product>
        <product>Latches - Hasps </product>
        <product>Multi-Point Handles and Latches </product>
        <product>Non-Skid Abrasive </product>
        <product>Quarter Turn Latches </product>
        <product>Rubber Feet </product>
    </products>
    <services>
        <service>Engineering Application Assistance</service>
        <service>Reusable Packaging </service>
        <service>Kitting</service>
        <service>MRP Programs </service>
        <service>Electronic Data Interchange (EDI) </service>
        <service>Same Day Shipments </service>
        <service>In-Plant Process Management</service>
        <service>Product Engineering Capabilities </service>
        <service>JIT/Kanban System </service>
        <service>Potential In-Plant Personnel </service>
        <service>Monthly Summary Invoicing </service>
        <service>Consolidated Deliveries </service>
        <service>Electronic Funds Transfer </service>
        <service>Parts Standardization </service>
    </services>
    <datevisited>09th August 2004</datevisited>
    <creator>Swathy Giri</creator>
</company>

```

Figure 12: Sample XML document DS2

4.2. Evaluation metric

We evaluated the classifier accuracy by comparing the classifier results for each test document with ‘truth’. The category from which the test document was selected should, ideally, be the category assigned to it by the classifier. The classifier produces a list of category id’s and weights, sorted in decreasing order of weights. Since the total number of categories in our experiment is 4, this list can contain a maximum of 4 values. The evaluation algorithm compares the truth-value for each document to the classifier result and finds the position at which the truth-value appears. Our evaluation metric calculates the percentage of test documents for whom the classifier has the truth-value as the top match, and in 2nd, 3rd and 4th. These values are represented cumulatively, reporting 100% for the 4th place (assuming all documents are classified).

4.3. Evaluation experiments with DS1

The goal of this experiment was to evaluate the effect of different XML fields on classification accuracy. In order to evaluate the hypothesis that some fields provide better classification than others, we compared the classification accuracy of a classifier trained on individual fields with one that ignored the XML markup altogether. Thus, the baseline for our experiments was a classifier trained and tested on the full-text documents. Once we had determined the relative accuracy of classifiers using individual fields, we then evaluated weighted combinations of the fields.

4.3.1. Experiment 1- Classification with single fields

Hypothesis: Classifying based on the contents of certain fields individually will provide better results than classifying on the contents of the whole document, and fields will vary on how well they can be used for classification.

Procedure: We classified the test documents (10 documents per category) from DS1 using classifiers trained on contents of each of the fields (30 documents per category from DS1) separately. The results obtained from the classifier were compared to our baseline system.

Setting up a baseline: We chose the vector space categorizer developed for the KeyConcept (KC) project [9] as our baseline system. The KeyConcept indexer was trained using the training documents from DS1 (30 per category) after which test documents from DS1 were classified. Full-text documents were used for training and classification for the baseline.

Results: Table 1 summarizes the results for each classification run and figure 13 provides a graphical representation of the results.

Fields	ALL	Individual Fields							
		Pub date	Copyright	Creator	Link	Title	Language	Description	Details
% matches at #1	90	25	25	27.5	27.5	55	65	70	92.5
%matches at # 2	97.5	65	52.5	50	52.5	72.5	65	90	97.5
%matches at # 3	97.5	77.5	72.5	75	72.5	77.5	67.5	97.5	97.5
%matches at # 4	100	100	97.5	100	97.5	80	100	100	100

Table1: Classification accuracy for full-text vs individual fields

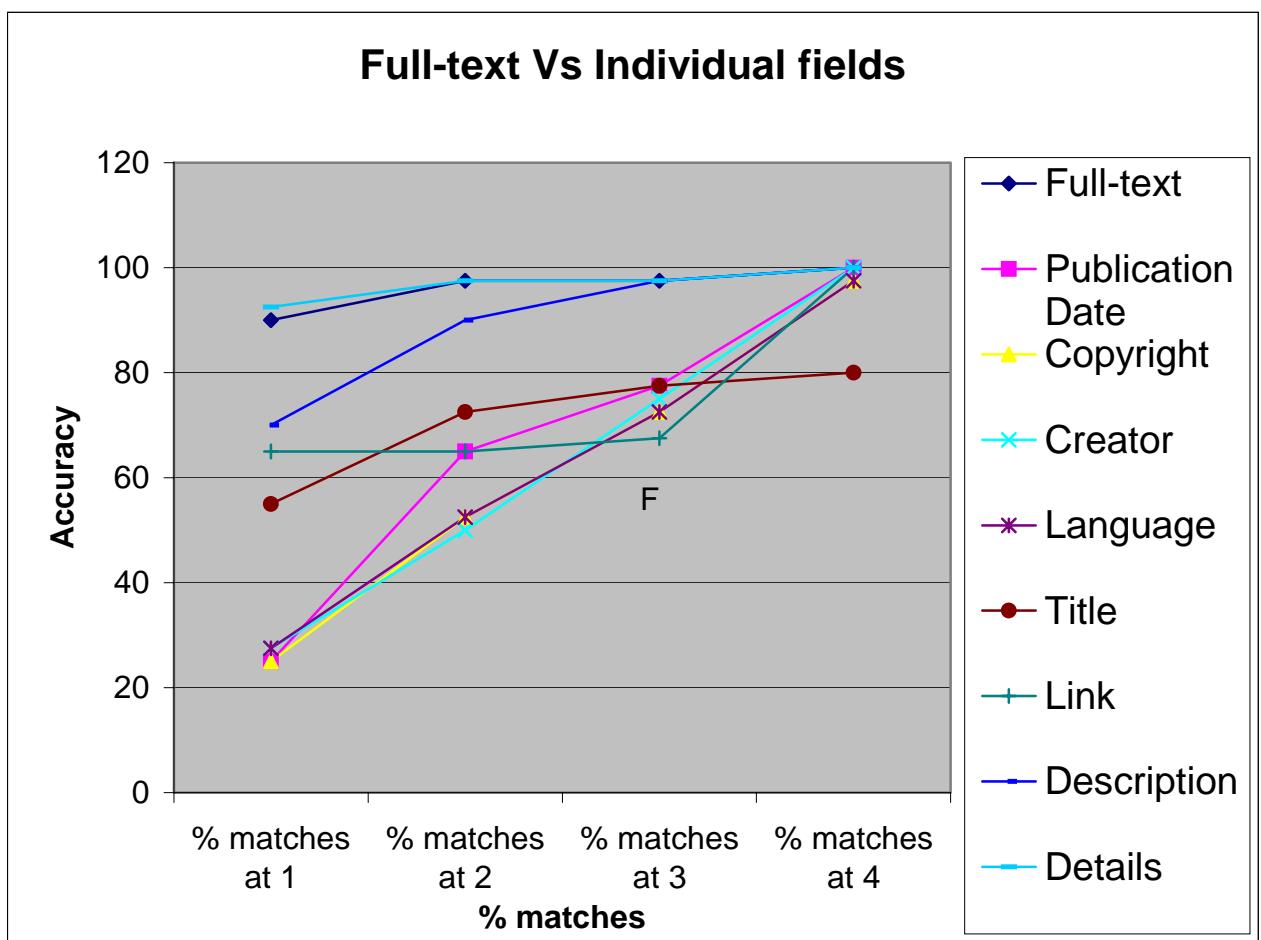


Figure 13: Classification accuracy for full-text vs individual fields

Discussion: From this, we conclude that a single field, Details (92.5% accuracy), performs as well or better than the full-text of the document (90%). We observe great variability between fields. The worse performing fields, for example, Publication Date, Language, and Copyright, produce only 25% accuracy at the top match. The Details field has a large number of tokens, the highest number of tokens compared to all other fields in DS1, variability in the content from one document to another, and a very low percentage of numbers compared to text. In contrast, fields that yielded poor classification results contained a higher ratio of numbers/date, or had little or no variability in their contents. The Title, Link and Description fields did not perform as well as the Details fields, since they had fewer tokens, however they did better than the other fields and we relate this to variability in their content.

4.3.2. Experiment 2- Classification with combination of fields

Hypothesis: Classifying documents based on a combination of fields can yield a better result than classifying documents on a single field separately. Weighting fields differently can also improve classification.

Procedure: To verify this, we generated all possible combinations of field weights ranging 0.0 to 1.0 in steps of 0.2, that summed up to a total weight of 1.0 were generated. Each test document was then classified by each non-zero weighted classifier, and the results were combined using the fusion module discussed in Section 3.2.3.

Setting up a baseline: We used the same vector-space classifier as in Experiment 1 as a baseline for this experiment.

Results: Table 2 shows the evaluation of the top matches from the results obtained for the experiment. Combinations that provided the highest accuracy among all combinations with 2, 3, 4 and 5 fields have been listed. We did not examine combinations of more than 5 fields because this would approach the baseline.

# non-zero fields	None	1 field	2 fields	3 fields	4 fields	5 fields
% accuracy at top match	90%	92.5%	92.5%	95%	95%	87.5%

Table 2: Results of experiment 2 on DS1

We will now present the data in more detail, for each number of fields used. Table 3 shows all the weight combinations that produced the highest accuracy (92.5%) for 2 fields. Similarly Table 4 shows all the weight combinations that produced the highest accuracy (95 %) with 3 fields, Table 5 shows all the weight combinations that produced the highest accuracy (95 %) with 4 fields and Table 6 shows all the weight combinations that produced the highest accuracy (87.5 %) with 5 fields.

PubDate	Copyright	Creator	Link	Title	Language	Description	Details
0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.8
0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.8
0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.8
0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.8
0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.8

0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.6
0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.2
0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.6

Table3: Combinations with 2 fields providing highest accuracy

PubDate	Copyright	Creator	Link	Title	Language	Description	Details
0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.6
0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.6
0.0	0.2	0.4	0.0	0.0	0.0	0.0	0.4

Table4: Combinations with 3 fields providing highest accuracy

PubDate	Copyright	Creator	Link	Title	Language	Description	Details
0.0	0.0	0.2	0.2	0.0	0.2	0.0	0.4
0.0	0.2	0.0	0.0	0.0	0.2	0.2	0.4
0.0	0.2	0.0	0.2	0.0	0.2	0.0	0.4
0.0	0.4	0.0	0.2	0.0	0.2	0.0	0.2
0.0	0.4	0.2	0.0	0.0	0.2	0.0	0.2

Table5: Combinations with 4 fields providing highest accuracy

PubDate	Copyright	Creator	Link	Title	Language	Description	Details
0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
0.0	0.0	0.2	0.0	0.2	0.2	0.2	0.2
0.0	0.0	0.2	0.2	0.2	0.0	0.2	0.2
0.0	0.2	0.0	0.2	0.2	0.0	0.2	0.2

0.0	0.2	0.2	0.0	0.2	0.0	0.2	0.2
-----	-----	-----	-----	-----	-----	-----	-----

Table6: Combinations with 5 fields providing highest accuracy

Discussion: The results show that combinations of 3 and 4 fields produced the best results. Further analysis reveals that the combinations that included the Details field with one or more of Title field, the Description field and/or the Link field performed well. Adding the other fields did not contribute to improving the classification.

4.3.3. Conclusions

In order to be able to select and weight fields for XML documents where truth is not known, we need a mechanism to identify useful fields *a priori*. From our observations, we list the following characteristics of well-performing fields:

- Fields that have a large number of tokens should be included.
- Fields with higher variability in their content should be included.
- Fields that have dates/numbers in their content are not helpful for classification.

Chapter 5 provides a detailed description of an algorithm that can be used to predict valuable fields for classification.

Chapter 5-Predicting Valuable Fields for XML Classification

The goal of this chapter is to perform a detailed analysis of the content of fields in DS1 and design an algorithm based on the characteristics of fields that improved classification results. We want to see if this algorithm can be used to predict valuable fields for a previously unseen document collection, producing better results for classification than that obtained from a full-text classifier. Section 5.1 provides a detailed analysis of the field characteristics for DS1, Section 5.2 evaluates the analysis results from Section 5.1 on a previously unseen data set DS2, Section 5.3 outlines an experiment performed on DS2 to verify the predictions, and finally Section 5.4 discusses how close the results were

to our predictions.

5.1 Analysis of DS1

We performed a detailed analysis of the contents of the fields in DS1 in order to design an algorithm that can be used to choose the most valuable fields and their weights, given any new XML data set. From the results we obtained from our experiments in Chapter 4, we found that fields Link, Title, and Details when combined with weights 0.2, 0.2, and 0.6, provided the highest accuracy for classification. We have used this information to design an algorithm that would predict these fields and weights for DS1. Table 7 provides the analysis of each of the fields. We considered 3 different characteristics for each of the fields across all the documents in the collection DS1, i.e. the number of tokens, variability in content, and the percentage of numbers.

Row	Characteristics	PubDate	Copyright	Creator	Link	Title	Language	Description	Details
1	# of tokens	242	26	241	311	1554	5496	6319	135929
2	Normalized Score # of tokens	0.0016	0.0002	0.0016	0.0021	0.0104	0.0366	0.0421	0.9055
3	Variability	0.128	0.077	0.008	0.013	0.714	0.075	0.378	0.172
4	% of numbers	37.23%	0.00%	0.00%	0.00%	0.64%	18.20%	0.79%	0.52%
5	# tokens Score	0.01	0.00	0.01	0.02	0.08	0.29	0.34	7.24
6	Variability Score	4	3	1	5	8	2	7	6
7	% of numbers Score	1	8	8	8	4	2	3	5

8	Total Score	5.04	11.00	9.04	13.05	12.25	4.88	11.01	32.73
9	Relative Score	0.05	0.11	0.09	0.13	0.12	0.05	0.11	0.33
10	Weights	0	0	0	0.2	0.2	0	0	0.6

Table 7: Analysis of characteristics of fields in DS1

We determined the characteristics by using the following formulae:

of tokens in T_i = Total number of words in Field T_i for all documents. (1)

Variability for T_i = Number of unique words in T_i for all documents (2)

$$\frac{\# \text{ of tokens in } T_i}{\# \text{ of tokens in } T_i}$$

% of numbers in T_i = $\frac{\text{Total number of numbers in } T_i}{\text{Total number of characters in } T_i} \text{ (3)}$

$$\frac{\text{Total number of characters in } T_i}{\text{Total number of characters in } T_i}$$

Once we obtained these values for all the fields in the collection, we assigned a score to each characteristic for every field in the range of 1..8 (the number of tags). The scores were assigned as follows:

of tokens score - We calculate the % of tokens that occur in a given field and multiply that by the number of fields.

$$\% \text{ of tokens } T_i = \frac{\# \text{ of tokens for field } T_i}{\# \text{ of tokens for field } T_i} \text{ (4)}$$

$$\sum_{i=1}^n \# \text{ of tokens } T_i$$

After we obtained the % of token score, we multiplied it by 8 (# of fields in DS1), to obtain the overall score for # of tokens.

$$\text{i.e., } \# \text{ of tokens score for } T_i = \% \text{ of tokens } T_i * 8 \quad (5)$$

Variability score- We rank ordered the fields by variability and scored the most variable field as “8” and the least variable field as “1”. Thus, higher variability led to a higher score. (6)

% of numbers score- Since we would like to exclude fields with a higher % of numbers, we rank ordered the fields by their % of numbers and scored the highest percentage as “1” and the lowest as “8”. (7)

$$\text{Total score for } T_i = 3 * \# \text{ of tokens score} + \text{Variability score} + \% \text{ of numbers score} \quad (8)$$

Since the results from Experiment 1 on DS1 showed that the best results occurred when the Details field was highly weighted, and Details is the clear winner on the # of tokens score, we have weighted the # of tokens score 3 times more than the other fields

$$\text{Relative score for } T_i = \frac{\text{Total Score for } T_i}{\sum_{i=1}^n \text{ Total score for } T_i} \quad (9)$$

The fields that gave the best result for DS1 were Details, Link and Title. Thus, we fixed a

threshold (TH) value using the formula,

$$\text{TH (for relative score)} = \frac{\sum_{i=1}^n \text{Total score for } T_i}{\# \text{ of tags} * 100} \quad (10)$$

that gave us a value of 0.120 for the relative score for field selection since this would yield our desired set of fields. We next need to assign weights to the fields. To do this we re-normalize using the total score of the selected fields, rather than all the fields.

$$\text{Weight } T_i \text{ (for fields with Relative score above TH)} = \frac{\text{Relative score } T_i}{\text{Sum of Relative scores of fields above TH}} \quad (11)$$

Weight T_i (for fields with Relative score below TH) = 0

Finally, all the weights were rounded to one digit precision. Applying the above procedure to DS1 gave us the following result:

Link, Title and Details are the valuable fields for classification with weights 0.2, 0.2 and 0.6. This is the set of weights that produced the best results for Experiment 2 for DS1. Thus this algorithm can “select” weights for accurate classification for DS1.

We can summarize the above procedure to obtain an algorithm that can be used to predict valuable fields and weights, for any given data set as follows:

5.1.1 Algorithm to predict valuable fields:

Step1: Calculate # of tokens, Variability, and % of numbers for each field across all the documents in the collection using formulae 1,2 and 3 respectively.[Rows 1,3,4 of Table 7]

Step2: Calculate Normalized # of tokens score by using formula 4.[Row 2, Table 7]

Step3: Calculate # of tokens score, Variability score and % of numbers score by using formulae 5,6 and 7 respectively.[Row 5,6 and7, Table 7]

Step4: Calculate Total score and Relative score using formulae 8 and 9 respectively.[Rows 8,9, Table 7]

Step5: Calculate the threshold (TH) value using formula 10, and apply it to the Relative score of every field to determine whether or not the field will be included for classification.

Step6: Finally, calculate the Weights for each field with Relative score above TH using formula 10 and assign 0 weights to all other fields.[Row 10, Table 7]

5.2 Evaluating DS2 with Algorithm from Section 5.1.1

The algorithm described in the previous section was designed with knowledge of what fields produced good results. It now remains to show that the same algorithm can be used to predict field weights for a new data set that produces higher classification accuracy than full-text classification. Applying the algorithm from Section 5.1.1 to DS2, predicts fields Product and Service, each with a weight of 0.5. The detailed analysis of the characteristics of DS2 is shown in Table 8.

Characteristics	Name	url	HQ Location	BR Location	Product	Service	Date Visited	Creator	HQ Phone	BR Phone
# of tokens	385	550	914	99	2134	759	360	240	338	34
Normalized Score # of tokens	0.0662	0.0946	0.1572	0.0170	0.3671	0.1306	0.0619	0.0413	0.0581	0.0058
Variability	0.618	0.305	0.639	0.919	0.568	0.623	0.058	0.008	0.763	1.000
%of numbers	0.23%	0.22%	24.67%	19.40%	0.47%	0.13%	44.44%	0.00%	99.11%	100.00%
# tokens Score	0.53	0.76	1.26	0.14	2.94	1.04	0.50	0.33	0.47	0.05
Variability Score	5	3	7	9	4	6	2	1	8	10
% of numbers Score	7	8	4	5	6	9	3	10	2	1
Total Score	13.59	13.27	14.77	14.41	18.81	18.13	6.49	11.99	11.40	11.14
Relative Score	0.10	0.10	0.11	0.11	0.14	0.14	0.05	0.09	0.09	0.08
Weights	0	0	0	0	0.5	0.5	0	0	0	0

Table 8: Analysis of characteristics of fields in DS2

Where,

HQ- Head quarters and **BR-** Branch

5.3. Experiment 3- predicting the valuable fields for DS2

Hypothesis: Combining fields Product and Service each with weight 0.5, as predicted by our algorithm, will yield a better result for classification than a full-text classifier for

DS2.

Procedure: To verify if our predictions were correct, we classified the test documents from DS2 using classifiers trained on the fields Product and Service separately and combined the results with the weight of 0.5 for each field using the fusion module as discussed in Section 3.2.3.

Setting up a baseline: We chose the same vector space categorizer as our baseline system. The indexer was trained using the training documents from DS2 (30 per category), after which 10 test documents from DS2 were classified. Full-text documents were used for training and classification for the baseline.

Results: The combination predicted by our algorithm yielded an accuracy of 82.5 % whereas our full-text baseline system yielded an accuracy of 65 %.

Discussion: From the results, we conclude that our system performs 25 % better (17.5 % absolute improvement) than our baseline system. For comparison, we used the brute force method to find which fields provide the highest accuracy for DS2 and we found that the Product and Service combination with weights 0.6 and 0.4 respectively (or vice versa) yielded the highest accuracy of 85 %. Thus our prediction algorithm performed within 0.03 % (2.5 % absolute) of the best combination found by the brute-force method.

Conclusions: The results show that by using our algorithm for predicting valuable fields and their weights, we obtained an accuracy of 82.5 %, which is considerably higher than our baseline (65%) and very close to the best accuracy (85%). Thus, our hypothesis is true and our algorithm can be used to predict combination of fields and their weights that will provide an improved accuracy.

Chapter 6- Conclusions and Future Work

6.1. Conclusions

We have presented the idea of content-based classification for XML documents and have shown that, due to the characteristics of the content of fields, some can be used to improve classification over that achievable with full-text. We identified characteristics of the most useful fields and developed an algorithm that can be used to predict useful fields and their weights for new XML data sets.

6.2. Future Work

Although the thesis provides a good insight to the classification of XML documents using a content-based approach, there are many avenues that can be explored to further improve classification of XML documents. A few possible suggestions are discussed in this section.

6.2.1 Extension to multiple Schemas

This system currently works on a single XML schema but it could be extended to work on multiple schemas. One possible approach to achieve this would be to normalize all the participating schemas to a generic schema and then perform classification. We could also try to include Tag structure along with content for classification.

6.2.2 Automating field selection

Although we have provided an algorithm that predicts the fields that would be useful for classification and their weights, we have not implemented it. We have key pieces for this algorithm implemented such as calculating the # of tokens, % of numbers and variability, that can be used to automate the entire algorithm.

6.2.3 Further Validation

The data sets we have used in our project are not large. So to further validate our results, larger data sets can be used. Also data sets that have a larger schema can be used since the schema for the data sets we have used for testing had a maximum of 10 tags. We were unable to find real XML data sets for validation. Hence, testing our algorithm with a real world data set also remains as part of the possible future work.

References

[1] N. Fuhr and G. Weikum. **Classification and Intelligent Search on Information in XML.** *Bulletin of the IEEE Technical Committee on Data Engineering*, 25(1), 2002.

[2] Attardi, G., Gull'ý, A., Sebastiani, F. **Automatic Web page categorization by link and context analysis.** In Hutchison, C., Lanzarone, G., eds.: *Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence, (Varese, IT)*

[3] Yang, Y. and Liu, X. **A Re-examination of Text Categorization Methods.** *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Berkeley, CA, 1999, 42-49.

[4] Guo G, Wang H, Bell D, Bi Y and Greer Y (2004): **Using kNN Model for Automatic Text Categorization**, *Journal of Soft Computing*, Springer-Verlag Heidelberg.

[5] McCallum, A. and Nigam, K. **A Comparison of Event Models for Naive Bayes Text Classification.** in *AAAI-98 Workshop on Learning for Text Categorization*, Madison, WI, 1998, 41-48.

[6] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger. **Tackling the Poor Assumptions of Naive Bayes Text Classifiers.** *Proceedings of the Twentieth International Conference on Machine Learning*, 2003

[7] H. T. Ng, W. B. Goh, and K. L. Low. **Feature selection, perceptron learning, and a usability case study for text categorization.** In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67-73, 1997

[8] Janez Brank; Marko Grobelnik; Nataša Milic-Frayling; Dunja Mladenic. **Training text classifiers with SVM on very few positive examples**

[9] Susan Gauch, Juan M. Madrid, Subhash Induri, Devanand Ravindran, and Sriram Chadlavada. **KeyConcept: A Conceptual Search Engine**, *Information and Telecommunication Technology Center, Technical Report: ITTC-FY2004-TR-8646-37*, University of Kansas.

[10] **Introduction to modern Information Retrieval** (G. Salton, MJ McGill, McGraw-Hill)

[11] F. Sebastiani. **Machine learning in automated text categorization.** *ACM Computing Surveys*, 34(1):1–47, 2002.

[12] **XML BASICS** -<http://www.softwareag.com/xml/about/starters.htm>

[13] **XML Basics and Benefits-**
http://www.intranetjournal.com/articles/200312/ij_12_08_03a.html

[14] Roy Goldman, Jason McHugh, Jennifer Widom. **Semistructured Data to XML: Migrating the Lore Data Model and Query Language**, Stanford University

[16] **Knowledge Management**, IBM India Research Lab

<http://www.research.ibm.com/irl/projects/knowledge.html>

[17] Mark Craven, Dan Dipasquo, Dayne Freitag, Andrew K McCallum , Tom M.Mitchell, Kamal Nigam , and Sean Slattery. **Learning to extract symbolic language from the WWW**. In *proceedings of AAAI-98, 15th Conference of American Association for Artificial Intelligence*, pages 509-516, Madison , US, 1998.

[18] L. Denoyer and P. Gallinari. **A belief networks-based generative model for structured documents. An application to the XML categorization**. In *MLDM 2003*, 2003.

[19] Yiming Yang and Christopher G. Chute. **An application of least squares fit mapping to text information retrieval**. In *Proceedings of the ACM SIGIR*, pages 281--290, Pittsburgh, PA, June 1993. 163

[20] Mohammed J. Zaki, Charu Aggarwal, **XRULES: An Effective Structural Classifier for XML Data**, *9th International Conference on Knowledge Discovery and Data Mining*, Washington, DC, August 2003.

[21] B. Liu, W. Hsu, Y. Ma. **Integrating Classification and Association Rule Mining.**
SIGKDD, 1998.