A classifier for semi-structured documents

Jeonghee Yi*
Computer Science, UCLA
405 Hilgard Av.
LA, CA 95 90095
jeonghee@cs.ucla.edu

Neel Sundaresan*
NehaNet Corp.
San Jose, CA 95131
nsundare@yahoo.com

ABSTRACT

In this paper, we describe a novel text classifier that can effectively cope with structured documents. We report experiments that compare its performance with that of a well-known probabilistic classifier. Our novel classifier can take advantage of the information in the structure of document that conventional, purely term-based classifiers ignore. Conventional classifiers are mostly based on the vector space model of document, which views a document simply as an n-dimensional vector of terms. To retain the information in the structure, we have developed a structured vector model, which represents a document with a structured vector, whose elements can be either terms or other structured vectors. With this extended model, we also have improved the well-known probabilistic classification method based on the Bernoulli document generation model. Our classifier based on these improvements performes significantly better on pre-classified samples from the web and the US Patent database than the usual classifiers.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data Mining*

Keywords

Classification, Semi-structured document, Structured vector

1. INTRODUCTION

As semi-structured texts, such as texts in HTML and XML (eXtensible Markup Language) [2], proliferate, there is pressing need to support efficient and effective information retrieval (IR), search, and filtering on them. Text classification has been extensively studied in the IR community [14]. Conventional classifiers do not perform well on structured documents [6, 8]. The conventional classifiers perform poorly because they are designed for non-structured data. Document structures pose new challenges to automatic classifiers. We believe document structures contain high-quality semantic clues that purely term-based classifiers cannot take advantage of.The conventional text classifiers are designed for self-contained, flat (or non-structured) texts. Markups and formatting cues in the structured documents can mislead the classifiers; but their removal means that only partial information fed into the classifiers.

In this paper, we propose a novel classifier for semi-structured documents. First, we extend the conventional model of document. Instead of viewing a document as a bag of terms, we view it as a *structured vector* whose elements may be either simple terms or other structured vectors. We show path expressions of terms are equivalent to the *structured vector* representing the document. Second, we extend the probabilistic document classification model to obtain one suitable for structured vectors.

To our knowledge, 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. The new classifier significantly excel conventional text-based classifiers. It cuts down the rate of the classification error on US Patent data[1] from 70% to 17%, and the one on the sample data from Yahoo[2] from 67% to 40%.

The rest of this paper is organized as follows: in section 2, we review the basics of the conventional text classifiers. In section 3, we describe our new classifier. In section 4, we report the performance of the new classifier on test datasets. We review related work in section 5 and provide concluding remarks in section 6.

2. BACKGROUND: OVERVIEW OF TEXT CLASSIFICATION

Given a text document, a text classifier assigns one of the predefined class categories to it or determines the likelihood that the document belongs to them. A classifier is first provided a topic set with sample *training* documents for each topic. Training documents are supplied with attached pre-assigned classes. The classifier develops models for the classes, a process also called *learning*. Later, presented with previously unseen documents, the classifier assigns the best matching classes. This is called *testing*.

[*] The work was carried out at the IBM Almaden Research Center.

[1] http://www.ibm.com/patents

[2] http://dir.yahoo.com/Business_and_Economy/Employment_and_Work/Resumes/

KDD 2000, Boston, MA USA
© ACM 2000 1-58113-233-6/00/08 ...$5.00

In this section, we review vector space model of documents and Bernoulli document generation model used by [4, 5, 6].

2.1 Vector Space Model

In conventional text classification, the object being classified is a self-contained, non-structured (or flat) document. The flat text document is often modeled as a vector of *terms* [11, 12]. Let D be a set of text documents, and W be a given lexicon of n terms. $\vec{W} = < w_1, ...w_n >$ is a vector of terms in W. A text document, $d \in D$, can be viewed as an n-dimensional vector, $\vec{d} = < d_1, ...d_n >$. If term w_i occurs in d, d_i is the number of occurrences of the term in the document; otherwise, d_i is zero.

2.2 Bernoulli Document Generation Model

A document d is generated by first picking a class. Each class c has an associated multi-faced coin. Its face represents a term t and has some success probability $f(c,t)$, that is the occurrence rate of t in c. Terms in d are generated by flipping the coin a given number of times (i.e., the desired number of words in d), and taking terms corresponding to the face of the coin. We define the following notations:

$$\begin{aligned}
n(d,t) &= \text{the no. of occurrences of term } t \text{ in document } d \\
n(d) &= \text{the no. of terms in } d \\
n(c,t) &= \sum_{d \in c} n(d,t) \\
n(c) &= \sum_{t} n(c,t) \\
f(c,t) &= \begin{cases} \frac{n(c,t)}{n(c)} & \text{maximum likelihood estimation} \\ \frac{n(c,t)+1}{n(c)+L(c)} & \text{with Laplace's law of succession} \end{cases}
\end{aligned}$$

where $L(c)$ is the size of the lexicon of class c. If document d belongs to class c, then the prior probability $\mathrm{P}[t|c] = f(c,t)$. Thus, given the assumption that the occurrences of terms in a document are independent,

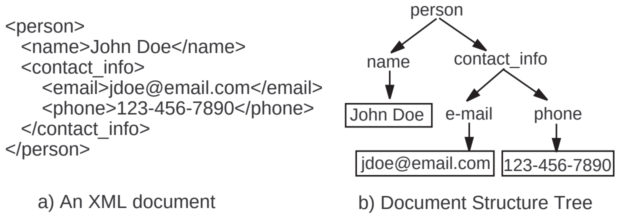
$$P[d|c] \quad = \quad \binom{n(d)}{\{n(d,t)\}} \prod_{t} f(c,t)^{n(d,t)} \qquad (1)$$

where $\binom{n(d)}{\{n(d,t)\}} = \frac{n(d)!}{n(d,t_1)! \cdot n(d,t_2)! \cdot \ldots}$.

3. SEMI-STRUCTURED DOCUMENT CLASSIFICATION

Semi-Structured data is data that does not have to conform to a fixed schema [1]. Semi-Structured documents are text files that contain semi-structured data. The examples are BibTex file, SGML (Standard Generalized Markup Language) [7], HTML, or XML. In this paper, we concentrate on XML documents. XML documents differ from typical text documents in the following respects:

1. Each element of an XML document is tagged.
2. The tags are usually nested, which makes XML documents hierarchical in nature.
3. Any element of an XML document can be referred to by any other elements which requires the documents to be modeled by a directed graph, rather than a tree.



```
<person>
  <name>John Doe</name>
  <contact_info>
    <email>jdoe@email.com</email>
    <phone>123-456-7890</phone>
  </contact_info>
</person>
```

a) An XML document

b) Document Structure Tree

Figure 1: A tiny XML document and its document structure tree.

Thus, proper classification of XML documents requires a scheme that exploits the above properties of XML. We extend the conventional document model in order to incorporate hierarchical sectioning of text. In the extended model, a document is hierarchically structured and text is embedded in the structure. For example, a book consists of many chapters that consist of sections. Each portion of text of a book belongs to a section, which in turn belongs to a chapter that contains the section, and to the book at its highest level.

We describe *structured vector model* and *semi-structured document generation model* in section 3.1 and 3.2, respectively. We present the tag augmentation method and semi-structured classifier in section 3.3 and 3.4, respectively.

3.1 Structured Vector Model

We have developed a document model for semi-structured documents, called *structured vector model*,based on the following observation: terms from the same XML element have to be grouped together i) to be treated together and ii) to be differentiated from terms in other XML elements. The reason is that terms from one substructure of a document may have a distribution different from that of terms in another substructure (or the distribution of terms in the entire document). The flat vector space model fails to encode this type of information.

Before we present the formal definition of *structured vector model*, we introduce the concept informally with example. Figure 1 presents a tiny XML document and its document structure tree. A document corresponds to the root element of its corresponding document tree. Each tagged element of root node corresponds to a child node of the root node in the document tree. In the example, `<name>` and `<contact_info>` are child nodes of `<person>` root element of the document. Likewise, any tagged element is a child of the higher level element in the nested structure. In the example, tagged element `<email>jdoe@email.com</email>` is a child of `<contact_info>` element. Pure text can not have any child, thus is represented as a leaf in the structure tree. Therefore, interior nodes of document structure tree correspond to tagged elements of document, and text is stored in leaf nodes of the tree. The label of a interior node is the tag of the corresponding document element.

For simplicity of exposition, we restrict attention to structures of documents that can be modeled by a tree, although, in general, they are better modeled by directed graphs. Ta-

Symbol	Meaning
t	a term
d	a document
e_d	structure tree of a document d
$e_d(0,0)$	the root element of a document
$e_d(i,j)$	j^{th} element node at level i of the structure tree
$m_d(i,j)$	# of child nodes of $e_d(i,j)$
$m_d(i)$	# of element nodes at level i of d
h_d	height of structure tree of d
$t_d(i,j)$	text node embedded in $e_d(i,j)$
$t_d(\vec{i},j)$	term vector of leaf node $t_d(i,j)$

Table 1: Symbols used in section 3

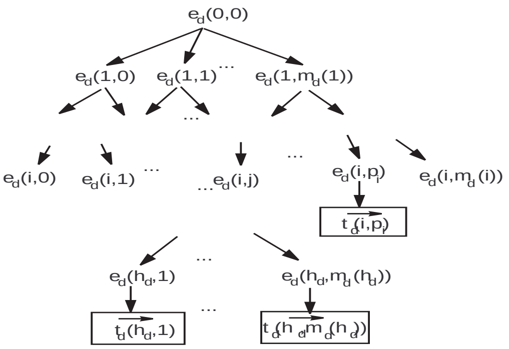


Figure 2: Structured Tree of a Document: an interior node, $e_d(i,j)$, corresponds to an element of the document, and a leaf node, ,$t_d(\vec{i},j)$, is a vector of document text terms that is embedded in $e_d(i,j)$. $m_d(i)$ is the number of element nodes at level i of the document and h_d is the depth of the structure tree.

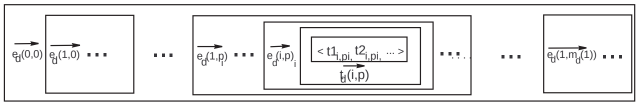


Figure 3: Structured Document Vector: vectors for substructures are nested in the document vector. The path to $t_d(i,p_i)$ is $(e_d(0,0).e_d(1,p_1).e_d(2,p_2)...e_d(i,p_i))$.

ble 1 lists the symbols used in the formal definition. Figure 2 shows the general structure of a document, d. $e_d(0,0)$ is the root element of the document in question and corresponds to root node of the structure tree. Each node at level $(i-1)$ of the tree is further refined by nodes at i^{th} level, $e_d(i,\cdot) = \{e_d(i,1), e_d(i,2), ..., e_d(i,m_d(i))\}$, until the leaves of the tree are reached. $m_d(i)$ is the number of nodes at level i of the document structure tree. A path to an interior node $e_d(i,j)$, $p_d(i,j) = (e_d(0,0).e_d(1,p_1).e_d(2,p_2)...e_d(i,j))$ is a sequence of structure nodes that must be visited to reach to the node $e_{i,j}$ from the root. The path from the root to the node $e_d(i,j)$ is unique, as each node has only one parent node. Thus, $p_d(i,j)$ uniquely determines the location of the corresponding element's location in the document. In the rest of this paper, sometimes we use $e_d(i,j)$ and $p_d(i,j)$ interchangeably when it does not cause any confusion. Each leaf node contains text terms of the element that immediately contains the text.

Alternatively, a document is represented by a vector, $e_d(\vec{0},0)$ $= < e_d(\vec{1},0), e_d(\vec{1},1), ..., e_d(1,\vec{m}_d(i)) >$. The vectors are defined recursively: $e_d(\vec{i},j)$ is a vector that consists of all subvectors, $e_d(i \vec{+} 1,k)$ of its children elements, where $0 \leq k < m_d(i,j)$, $m_d(i,j)$ is the number of child nodes of $e_d(i,j)$. This results in the nested document vector shown in figure 3 that corresponds to the document in figure 2.

Documents that belongs to a class do not have to conform to a specific schema. We define the *document structure of a class* as a superset of the structures of all documents belongs to the class. The document class tree structure, E_c, consists of all document structures in the class, i.e. $E_c = \{p_c(0,0), ..p_c(i,0), ..p_c(i,m_c(i)), ..p_c(h_c,0), ..p_c(h_c,m_c(h))\}$ [3], where $m_c(i)$ denotes the number of nodes at level i of the class document tree and h_c denotes the height of the class document tree.

3.2 Document Generation Model

Let us define new notations:

$$\begin{aligned}
p_d(i,j) &: \text{a path to a node } e_d(i,j) \text{ from the root}\\
&= (e_d(0,0).e_d(1,p_1).e_d(2,j_2)...e_d(i,j))\\
p_c(i,\cdot) &= \{p_d(i,j) \mid j = 1,...,m_d(i)\}\\
n(d,p_d(i,j),t) &= \#\text{ of occurrences of } t \text{ in } p_d(i,j) \text{ of } d\\
n(d,p_d(i,j)) &= \#\text{ of all terms in } p_d(i,j) \text{ of } d\\
n(d) &= \sum_{p_d(i,j) \text{ of } d} n(d,p_d(i,j))\\
n(c,p_d(i,j),t) &= \sum_{d \in c} n(d,p_d(i,j),t)\\
n(c,p_d(i,j)) &= \sum_{t} n(c,p_d(i,j),t)\\
n(c) &= \sum_{p_d(i,j)} n(c,p_d(i,j))
\end{aligned}$$

The following outlines the Bernoulli document generation model for semi-structured documents :

$$P[d|c] = \sum_{i=0}^{h} \sum_{p_d(i,j) \in p_d(i\cdot)} P[d|p_d(i,j),c] \cdot P[p_d(i,j)|c] \quad (2)$$

For any path of length i,

$$P[p_d(i,j)|c] = P[p_d(i{-}1,p_{i-1})|c] \cdot P[e_d(i,j)|p_d(i{-}1,p_{i-1}),c] \quad (3)$$

$$P[d|p_d(i,j),c] = \binom{n(d,p_d(ij))}{\{n(d,p_d(ij),t)\}} \prod_{t \in p_d(i\cdot)} f(c,p_d(i,j),t)^{n(d,p_d(i,j),t)} \quad (4)$$

where

$$f(c,p_d(i,j),t) = \begin{cases} \frac{n(c,p_d(i,j),t)}{n(c,p_d(i,j))} & : \text{maximum likelihood estimation}\\ \frac{n(c,p_d(i,j),t)+1}{n(c,p_d(i,j))+L'(c)} & : \text{with Laplace's Law of Succession} \end{cases}$$

$L(c)$ is the size of the lexicon of class c.

[3] Here we define class document tree as a set of paths rather than element nodes, in order to preserve the path information of the document that it was originated.

Note that equation 4 requires weaker independence assumption than the conventional classifier like the one in section 2.2 (equation 1). Equation 1 requires the independence assumption to be held for the entire terms of the document, whereas equation 4 requires the assumption to be held only in the structure node it belongs to. This is pragmatically much better. Experimental results in a later scetion show how this weaker independence assumption requirement helps improving the classification results.

3.3 Path Expression by Tag Augmentation

Notice that the same term that occurs in different XML elements may have different meaning. For example, in the following document, the meaning of “course” is different depending on what element the term belongs to :

```
<resume>
  <name>John Doe</name>
  <education>
    .... CS courses taken include ...
  </education>
  <hobby>Taking walk on golf course.</hobby>
</resume>
```

We take the path expression of each term in the document by augmenting the tags of the structure elements it belongs to. For example, the term “course” in education element yields “resume.education.course”, and the one in hobby yields “resume.hobby.course”. Terms in the same element share the same path, whereas the same term in different elements have different path.

The path expression is equivalent to the path of pre-order traversal of a document tree. Each subvector, $v_d(i,j)$, of structured vector of a document d corresponds to an element, $e_d(i,j)$, of the structure tree. The subvector, $v_d(i,j)$, is contained in one-level higher subvector, $v_d(i-1,p_{i-1})$, the vector corresponding to $e_d(i-1,p_{i-1})$, the parent element of $e_d(i,j)$. Therefore, the nesting hierarchy of the structured vector corresponds to the pre-order traversal of the tree structure. Thus the path expression and the nesting hierarchy of the structured vector are equivalent.

Based on this property, we can encode documents by path expressions of terms in a single flat vector. Thus we can enjoy both the benefit of structured model and the fast computation of a flat vector.

3.4 The Semi-structured Classifier

For classification of semi-structured documents, we choose the class that maximizes the following *a posteriori* class probability:

$$P[c|d] = \frac{\pi(c)\prod_{t\in p_d(i\cdot),t\in d} f(c,p_d(i),t)^{n(d,p_d(i),t)}}{\sum_{c'}\pi(c')\prod_{t\in p_d(i\cdot),t\in d} f(c',p_d(i),t)^{n(d,p_d(i),t)}} \quad (5)$$

where $\pi(c)$ is the prior distribution of t on the class c. Refer to the definition of the notations defined in section 3.2.

4. EXPERIMENTS

4.1 Datasets

We ran experiments with our classifier on the following two datasets:

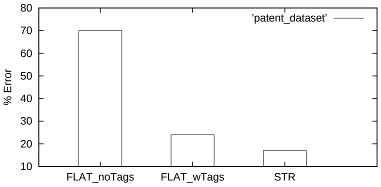


Figure 4: Comparison of classification error rate of *patent* datasets in three different settings. $FLAT_{noTags}$,$FLAT_{wTags}$, and *STR* achieved 70%, 24%, and 17% of error rates respectively.

1. US Patent Dataset

We downloaded 10,705 US patent documents in three categories from IBM's Patent Server [4]. We randomly split the documents into a training set of 7,136 documents and a test set of 3,569 documents. The documents are well categorized and the language is quite consistent. The XML is generated by a computer program with a common DTD. Thus, all documents share the same structure.

2. The *Résumé* Dataset

We downloaded about 1700 resumes of 52 topic areas from Yahoo[5]. From the resumes in HTML we eliminated content-irrelevant formatting directives, hyperlinks, images, and scripts and discarded about 450 documents that are mostly multimedia illustration. We further discarded or merged, if appropriate, topic areas with less than 30 resumes. For example, Biology, Chemistry, Earth Sciences, Mathematics, and Physics were merged to Science all together. Finally, we created a sample set of 890 documents of 10 topic areas, which was randomly divided into a training set of 600 documents and a test set of 290 documents.

4.2 Experiments on Patent Dataset

We compare the the performance of two algorithms: our new algorithm, called *STR*, and the classifier introduced in section 2 for flat document model, called *FLAT*. The classification errors were measured in 3 different settings:

- *STR*
- $FLAT_{noTags}$: *FLAT* without structure tags
- $FLAT_{wTags}$: *FLAT* with structure tags

The result is shown in figure 5: *STR*, $FLAT_{wTags}$, $FLAT_{noTags}$ achieved 17%, 24%, and 70% error rate, respectively. It is not so clear why $FLAT_{noTags}$ performs so badly, even in comparison to $FLAT_{wTags}$. Until more convincing explanation is found, we tentatively conclude the structure information itself contains valuable information on classification.

Note that *STR* did not perform its full potential due to the characteristics of the dataset. As shown in the equation 2 in section 3.2, *STR* takes into account both the distribution of structure element and the term distribution within the element. As discussed earlier, all patent documents share the same schema. Moreover, they conform to the schema quite rigidly. Since all documents have the same structure, equa-

[4] http://www.ibm.com/patents

[5] http://dir.yahoo.com/Business_and_Economy/Employment_and_Work/Resumes/

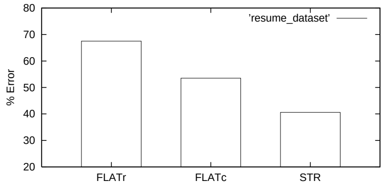


Figure 5: Comparison of classification error rate of *resume* datasets in four different settings. $FLAT_r$,$FLAT_c$, and *STR* achieved 67%, 53%, and 40% of error rates respectively.

tion 3 is the same for all documents. Thus, STR differs from $FLAT$ only in the computation of term distribution in various elements (equation 4). The improvement of error rates from 24% to 17% was achieved only by the difference in term distribution induced by document structure. This implicitly tells us that it would be great benefit to take into account the distribution of structure when documents structure in a corpus varies significantly.

4.3 Experiments on *Résumé* Dataset

For *resume* dataset, we compare classification errors of the two algorithms in three different settings:

- STR
- $FLAT_r$: $FLAT$ on raw HTML
- $FLAT_c$: $FLAT$ on HTML after cleansing

As shown in figure 5, STR, $FLAT_c$, $FLAT_r$ achieved 40%, 53%, and 67% error rate, respectively. Though the error rate of $FLAT_r$ seems high, comparable error rate was already reported by [6]. By using cleaned up HTML documents, $FLAT_c$ achieved 14% improvement over $FLAT_r$. STR improved 13% in comparison to $FLAT_c$. In fact, the scale of error reduction by STR is much greater considering the signal-to-noise ratio of terms. By tag augmentation, the number of terms used by STR is about 2.7 times more without adding any more training set. Therefore, with the same signal-to-noise ratio, this scheme has higher potential to perform better.

5. RELATED WORK

The classification problem has been addressed in statistical decision theory, machine learning, and data mining. Decision tree is a well-known machine-learning technique often used to classify numerical and categorical data [3, 10]. In general, the classifiers for numerical and categorical data handle data with much lower dimensionality in comparison to text. [9] applied decision tree algorithms on text. Neural network approach is applied to text categorization by [13]. Probabilistic classifiers are also used in text classification. To make the computation simple, *Naive Bayes* typically assumes that the word occurrence is independent. [9] reported the evaluation results of *Naive Bayes* on Reuters. TAPER [4] is a variation of *Naive Bayes* to build a classifier for hierarchical taxonomy.

All text classifiers discussed above assume that documents being classified are non-structured self-contained documents. [6] is a classifier that is designed specifically for hypertext such as HTML utilizing both local and non-local information (through hyperlink) to the document. The paper shows the classifiers designed for self-contained documents perform poorly on hypertext because the web documents are extremely diverse. However, [6] still does not take into account the structure of the document which also contains valuable information for classification.

6. CONCLUSION

We have developed a novel algorithm of classifying semi-structured documents by extending the underlying document model and using structure-based. The new classifier takes advantage of the information latent in the document structure as well as the text. Our experiments prove that the method improves the accuracy of the classification. For a set of US patent documents, the new method cuts down classification error from 70% to 17%. As next steps, we plan to explore context-sensitive feature selection on the basis of document structure.

7. REFERENCES

[1] S. Abiteboul. Querying semistructured data. In *Proc. of the International Conference on Database Theory*, pages 1–18, Delphi, Greece, Jan. 1997.

[2] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0, W3C Recommendation.* World Wide Web Consortium, Feb. 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth and Brooks/Cold, 1984.

[4] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Using taxonomy, discriminants, and signatures for navigating in text databases. In *Proc. of the 23rd VLDB Conference*, Athens, Greece, 1997.

[5] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. In *VLDB Journal*, 1998.

[6] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of ACM SIGMOD Conference*, pages 307–318, Seattle, Washington, 1998.

[7] C. Goldfarb. *The SGML Handbook. Y. Rubinsky, Ed.* Oxford University Press, 1990.

[8] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proc. of the 14th International Conference on Machine Learning*, Nashville, Tennessee, July 1997.

[9] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 1994.

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[11] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[12] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[13] E. Wiener, J. Pedersen, and A. Weigend. A neural network approach to topic spotting. In *Proc. of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.

[14] Y. Yang. An evaluation of statistical approaches to text categorization. In *Information Retrieval Journal*, May 1999.