A hybrid method to categorize HTML documents

M. Khordad, M. Shamsfard & F. Kazemeyni
Electrical & Computer Engineering Department,
Shahid Beheshti University, Iran

Abstract

In this paper we introduce a hybrid method for classifying HTML documents. In this method the statistical, semantic and writing style features of text are used to categorize documents. Categorization can be done in both supervised and unsupervised modes and categories may be predefined or be created dynamically (clustering). The classification system exploits an ontology of interesting topics. The ontology which contains categories and their hierarchical relations can be updated automatically during the system's lifetime. Newly defined categories can be added to the ontology and existing categories can be changed according to the documents received.

The statistical part of the method is based on the Rocchio algorithm. The algorithm has been changed to cover the special conditions for dynamic category building, for categorizing with and without training data and for variable length feature vectors. The semantic part of the algorithm exploits Wordnet to substitute words with their corresponding concepts and does some word sense disambiguation tasks prior to clustering. This way documents will be clustered according to their concepts instead of words. The other part of the method considers writing style features of text such as writing in bold/italic style, writing with different (bigger) fonts or occurring words and concepts in special places of the document, such as the title, headers or hyperlinks.

In this paper, after a brief overview on existing methods of document classification, the proposed method will be discussed and some experimental results of classifying documents will be shown. Experiments show that the hybrid method results in some improvements in performance (the accuracy).
Keywords: data mining, text categorization, clustering, Rocchio algorithm, ontology.

1 Introduction

Growing volume of electronic documents on the web results in a growing need to tools which help users to manage, search, filter and retrieve information from such a huge repository. One of the useful preprocessing steps to handle these requests is semantic and topical classification of documents on the web. Classification and categorization of web documents increases speed, correctness and performance of information retrieval in search engines. In addition text classification is used in applications such as natural language processing (NLP), data filtering, data analysis, and information security.

Text categorization is the problem of automatically assigning predefined categories to free text documents. In other words, the aim of text "categorization", "classification" and "clustering" is to find a topic or subject for a document with minimum error. The assigned topic may be predefined (categorization) or be created on demand (clustering). The classification task may de done with /without supervision. In supervised methods the classifier should be trained using labeled documents but in unsupervised methods not only there is no label but also the training set may be empty and all documents may be handled as the test set.

In this paper, we introduce a hybrid method for HTML page categorization, based on Rocchio [1] algorithm. The most important features of this method; are as follows:

- Combining statistical, semantic and writing style features of HTML pages,
- Using dynamic ontology and online creating or changing of categories,
- Multi-class categorization with / without supervision
- Variable feature space dimensions
- Dimensionality reduction by conceptual analysis and feature selection.

In the next section we will have a quick overview on some related works. Then we will discuss our proposed method, testing samples and the results of its evaluation.

2 Background

There are many different algorithms to classify text documents, each focusing on different features of text. These methods are quite different but they have a common framework. In most of these methods (especially statistical methods), at first, some features must be extracted from the document and then using a similarity measure the classifier should find the most similar category to the document according to that features. The difference between various methods is in feature selection, feature extraction, choosing similarity measure and preparing the training and testing data sets.

As selected features, Rocchio [1] and Joachims [2] use Frequency of word occurrences in text, Haav and Lubi [3] and Kuo and Wang [4] use document's hyperlinks, Ghasem-Aghaei and Sarafan [5] use natural language features and

keywords, Lu and Drew [6] use document's images, and Shamsfard *et al.* [7] use writing style of words and their location in the document for categorization.

After selecting features the categorization system may extract them by statistical or semantic (conceptual) methods. Semantic algorithms (as used by Haav and Lubi [3]) use natural language processing methods and retrieve semantic and conceptual features of text such as the meaning, linguistic constituents, conceptual relations and so on, while statistical methods exploit statistical properties such as word/term frequency, collocations and cooccurrences. Bayes algorithms, support vector machines (SVM), Rocchio method, decision tree algorithm and K-nearest neighbor are some statistical classification methods. In Naïve Bayes (used by Kuo and Wong [4]), conditional probability of words belonging to a special category is calculated. If this probability is greater than a threshold, then the document will be assigned to this category. In SVM algorithm (used by Joachims [8]) which is useful for binary classification tasks, a multi dimensional vector is produced for each document in which each dimension is related to a word. The weight of each dimension can be calculated by using the number of word occurrences in the current document and other documents in a related set. Rocchio algorithm is similar to SVM in producing document vector, but their difference is in calculating weights and similarity measures. Decision tree method (used by Gehrke *et al.* [9]) builds a tree whose nodes are documents terms and leaves are categories. In this method, classification is performed by searching the tree. In K-nearest neighbor algorithm (used by Han *et al.* [10]) all documents are treated as the points in an n-dimensional Euclidean space. Algorithm finds the k-nearest neighbors whose distance from test cases falls within a threshold p. A research by Ragas and Koster [11] compares some of these algorithms and shows that Rocchio has better results among the classification algorithms.

After extracting the features, they must be compared with the features of predefined categories. Different similarity measures and comparison methods are used to find the similarity between document's features and each category's features. For instance Rocchio [1] calculates cosine of the angle between document's feature vector and the category's feature vector, Han *et al.* [10] calculate Euclidean distance between documents and Haav and Lubi [3] compare document vector's length to find similarities.

Now the designer should prepare the training set. The classification may be done with or without supervision. In the first type the classifier usually uses a large labeled training data set. Most of classification methods such as the work done by Ragas and Koster [11] use two groups of documents for training containing positive and negative examples. This training style increases system's performance, but make training very difficult and slow. Some methods, like PEBL by Yu *et al.* [12], try to eliminate these limitations without performance falling. In PEBL, SVM algorithm has been changed to operate on only positive and unlabeled data. In unsupervised methods (clustering) the training data set is unlabeled and it may be empty or be entered to the system at once or incrementally.

3 The proposed method

The proposed classification method uses a combination of statistical, conceptual and writing style features of text to find the appropriate category for a HTML document. The statistical part of our method is based on Rocchio algorithm. To improve the performance of Rocchio algorithm we build the feature vectors for concepts instead of words. In other words each dimension of a document vector corresponds to a key concept in the document. On the other hand writing style of words (i.e. writing with specific fonts), word's location in the document (i.e. occurrence in the title or header) and also occurring in document's hyperlinks can affect the weight of the corresponding concept in the document vector.

In our method creation of topical categories can be done with or without supervision. In the supervised method, at first labeled documents (training set) are given to the system, and the category vector will be generated from the average of document's vectors. The category vector may be updated whenever new documents arrive. So, the test phase can be considered as the unsupervised training phase, too. On the other hand if the system explores a new category, it will be added to the ontology (unsupervised category creation) and its characteristics will be passed to user to make a name for the newborn category (supervised naming).

Figure 1 shows the structure of the HTML documents classification system. As it can be seen, HTML Parser, feature extractor, classification module, semantic analyzer and ontology manager are the main operating modules of this system. These modules use two essential knowledge bases: an ontology of topical categories and a semantic lexicon; WordNet [13].

In this section, we will describe the system's functionality in more details.

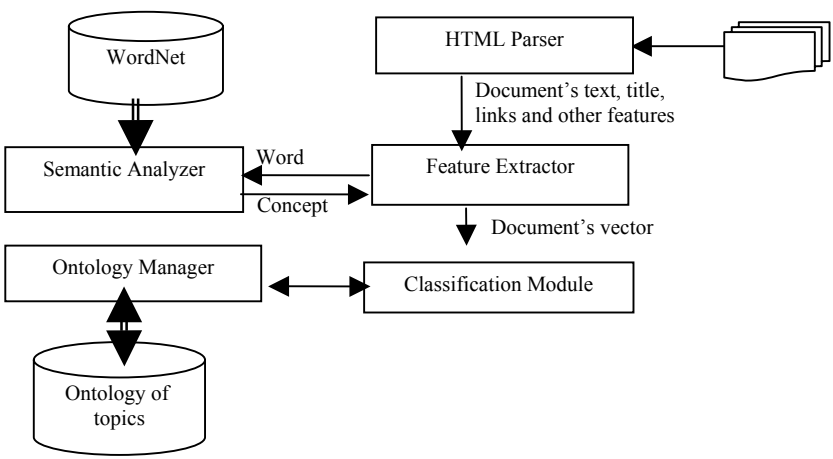

Figure 1: The structure of HTML categorization system.

3.1 Producing feature vectors

The HTML parser analyzes the HTML document, remove HTML tags and extract the text part of the document, its title, headers and hyperlinks. The output of the HTML parser will be fed to the feature extractor. The feature extractor is responsible for producing document's vector by analyzing statistical, semantic and writing style features of the document. In this vector each dimension belongs to the weight of a key concept in the document. Variable number of key concepts and therefore vector's dimensions for various documents is a useful property of our method. To find the key concepts, the first step is to delete superfluity words. These words can be recognized in two ways: First the stop words which are listed in a delete-list (such as prepositions, auxiliary verbs, pronouns, etc) and are common to all documents. And second the words which are unimportant for the current document, whose frequency in this document is less than a threshold or their frequency in whole documents is more than another threshold.

After removing these words, the semantic analyzer uses WordNet semantic lexicon to map the remaining words with their corresponding concepts. In the case that a word has more than one sense or meaning, we use a *Word Sense Disambiguation* (WSD) algorithm to find the most probable word sense. The WSD algorithm is based on word's definition (Lesk [14]). In this algorithm description fields of all word senses will be retrieved from WordNet. Then for each sense, all the words in the description field will be searched in the main document and their term frequency will be calculated. At last the word sense will be selected which has the maximum sum of these term frequencies.

At last the feature extractor assigns a weight to each concept using eqn (1).

$$d^{(i)} = TF(c_i, d) \times IDF(c_i) \times W(c_i) \tag{1}$$

$$IDF(c) = \log(\frac{|D|}{DF(c)})$$

where D is total number of documents, TF(c,d) is the term frequency of concept c in document d, DF(c) is the document frequency and shows the number of all documents in which concept c has occurred at least once and w(c) is the average weight factor of concept c in the text. Weight factor of each term (concept or word) determines the importance of this term based on writing style of text. For example this factor for terms in header or title of documents or bold or larger font (except at the beginning of a paragraph) is more than one and for others is one. As a concept may be related to more than one word or more than one occurrence of a word, its weight factor should be calculated by averaging.

After weighting all concepts, the ones whose weight is more than λ threshold are saved as key concepts and their total weights will be involved in the document vector.

The feature extractor will build feature vectors not only for each document (which we call document vector), but also for all topical categories in the ontology (which we call category vector). Category vector is an average of all document vectors related to each category in the training set.

It is noticeable that in our method the length of document vectors and also their dimensions (the key concepts) are different for different documents. So to

make the consistency, a category vector contains the union of concepts of its related document vectors with average of their weights.

3.2 Multi-class categorization of documents

To categorize a document the system should compare the generated document vector with category vectors generated from the training sets. Predefined topical categories are located in an ontology which currently presents just the hierarchical relations between different topics. Comparing document vector with category vectors in the ontology starts from the immediate sons of the root. Using cosine of angles (eqn (2)) as the similarity measure, the similar vectors can be found.

$$Cos(V_1,V_2) = \frac{\sum_{i=0}^{n} V_1[i] \times V_2[i]}{\sqrt{\sum_{i=0}^{n} V_1[i]^2} \times \sqrt{\sum_{i=0}^{n} V_2[i]^2}} \; if \; V_1[i] \neq 0 \tag{2}$$

In this equation, V_1 is the document vector and V_2 is the category vector. For analyzing this similarity there are two thresholds θ_1 and θ_2 ($0< \theta_2< \theta_1<1$) which are obtained from statistical processing of experiments (cf. section 4). If the cosine between document vector and category vector were more than θ_1 then document will be assigned to this category. If the cosine falls between θ_1 and θ_2, comparison will be continued between the document and this category's children in the hierarchy. And if the cosine is less than θ_2, a new node under the current node will be created as a new topical category. In cases which more than one category is found for a document, the most similar (with largest cosine) category will be chosen.

After assigning document to its category (new or old), the category vector should be updated by the ontology manager according to this new document.

3.3 Updating the ontology of subjects

Ontology manager updates the ontology of subjects. This process contains updating category vectors, adding new categories, deleting or merging categories or splitting a category to detailed categories. Updating category vectors is accomplished both in training and test phase. In both phases assigning a new document to a category, causes the category vector to be updated according to eqn (3).

$$d^{(i)}{}_{ct} = \frac{(d^{(i)}{}_{ct} \times N) + (d^{(i)}{}_{new} \times n)}{N + n} \tag{3}$$

where $d^{(i)}{}_{ct}$ is the i^{th} element of the category vector, $d^{(i)}{}_{new}$ is the i^{th} element of the new document vector, N is the number of documents assigned to this category and n is the number of documents forming the new document vector.

On the other hand the ontology would be verified and reorganized in specific periods. In this reorganization process, similar categories can be merged based on one of the following criteria:

First: *merge classes that the cosine of angle between their vectors is more than threshold* θ_1. Since assigning new documents to a category, changes its vector, it is probable that the similarity between categories be changed too. If this similarity becomes more than the merge threshold we can merge these classes. Checking the similarity between classes can be done in both breadth first and depth first traverses of the ontology tree. We used breadth first approach which is faster.

Second: *merge classes with the most common candidate documents*. In cases which we have found more than a category for a document, the similarity measure of these categories will be increased one point. When this similarity becomes more than the merge threshold, these categories will be merged.

4 Experimental results

The proposed method is implemented and tested on pages from different domains such as physics, business and TCPIP from computer field. Although the acquired document set is not large (about 100 documents extracted for each class), we performed many tests, in different environments and various parameters as follows:

Parameter adjusting – There are two types of parameters to be adjusted by tests; weights and thresholds. As we discussed, various writing styles have various weights in computing the feature vector. To find weights we have done some tests on collected documents. Figure 2 shows the results of some of them to find weight for words (concepts) in the header or title of the page. We expected that the best weight would maximize the rate of selecting suitable words from each document to participate in its vector.

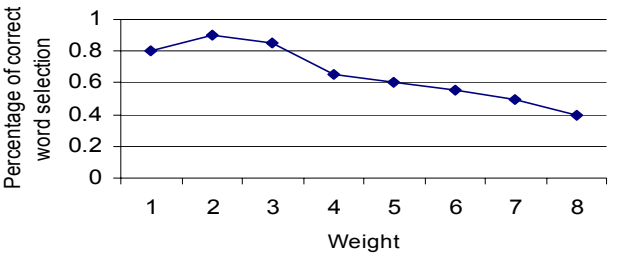

Figure 2: Selecting suitable words according to header and title weights.

As the figure shows the best weight for H1 (words in the title or first level headers) is about 2. According to same experiments we assigned 1.5 as the weight for bold or larger fonts and H3 (third level headers) and 1.05 as the weight of hyperlinks at this stage.

Thresholds for merging classes in ontology refinement and thresholds for accepting or rejecting the similarity between a document and a class are others to be adjusted by tests. We performed some tests on different thresholds to find the best. Our experiments showed that the best values for θ_1 is 0.6, for θ_2 is 0.4 (cf. section 3.2), and for the merge threshold (section 3.3) is 0.6 too.

Training set – To test the proposed method, the document collection has been divided into training set and test set repeatedly. On one side of this spectrum the training set is empty and all documents are in the test set and on the other side all documents are in both test and training sets. Between these two ends, the chosen collection is divided to different test and training sets (with no intersection). As figure 3 shows, increasing the percentage of documents in the training set, results in reduction of categorization error. The diagram is drawn for categorizing based on key concepts. In the best case which the system is learned by a large subset of documents (about 80% of them) the error (1-precision) is about %5. In the worse case which the training set is empty and the system performs an unsupervised clustering from scratch, the error is less than %30 and the average error rate is about %15.

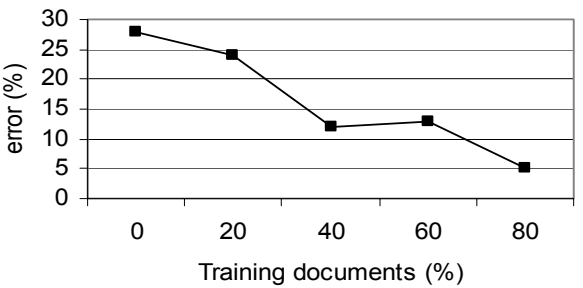

Figure 3: Categorization errors reduction.

Using concepts instead of words – To analyze the efficiency of using concepts instead of words, tests are done in two conditions: generating vector for key words and for key concepts. In the later case the WSD algorithm is used to find the appropriate concepts. Figure 4 shows the number of extracted concepts corresponding to the number of words in a document. It can be seen that the growth of concept numbers are less than word numbers.

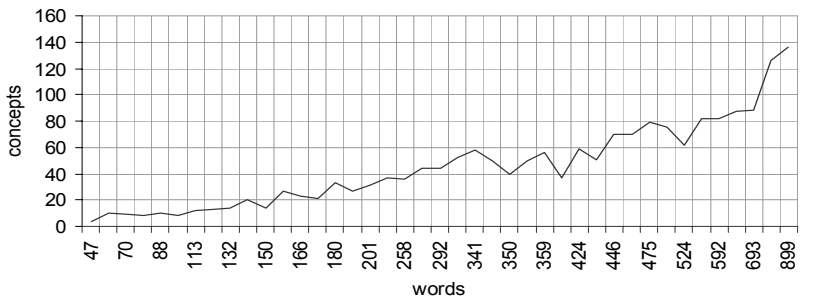

Figure 4: Number of extracted concepts in comparison with number of words in a document.

Results show that as it can be predicted, conceptualization does not improve the efficiency when there are not many synonym words or similar words (words with same root but inflectional differences) in the text. But in other situations this method improves the precision of categorization by %6.6 to %41. The various improvement rates relate to various tests with various styles of writing (more synonym or similar words, more improvements).

Dynamic constructions of the ontology – As the system may build new categories (especially while clustering) we reorganize the ontology of subjects periodically to merge similar categories and build more general ones. In different tests two introduced merge criteria were tested separately. The results show that each of them causes merging some similar classes. It was seen that composition of two methods can improve accuracy of categorization. In average the first criterion caused %16.6 of classes to be merged correctly and the second algorithm caused %38.8 of them to be merged correctly. In the worse case the error of merging (merging inappropriate classes) is reported about %3.2. Combining two criteria caused %44.4 of similar classes to be merged correctly. In our tests with current parameters, no splitting occurred while reorganizing the ontology. It means that the parameters force the algorithm to build categories as specific as possible.

5 Conclusion

We introduced a method for web documents categorization. In this method the efficiency and performance of an existing statistical method is improved using conceptual and writing style features. Another salient point in this method is dynamic construction of the ontology, document vectors and category vectors. This way (1) even after generating a category vector (training phase) arrival of a new document may cause the category vector to be changed (test phase), (2) reorganizing the ontology in predefined periods may result in merging or splitting classes and create new ones or delete the old ones.

To complete this effort following improvements are proposed as further works:

- Developing tests using larger training and test sets,
- Adjusting thresholds more accurately using more samples,
- Changing the ontology of topics to cover non hierarchical relations too,
- Improving the semantic analyzer to extract more conceptual relations from text, (e.g. attention to noun phrases instead of single words and increment the weights for their heads),
- Improving merging algorithms to cover a wider range of classes,
- Navigation of hyperlinks to increase the categorization efficiency,
- Exploiting deeper natural language processing methods to extract categorization information from text,
- Adding cohesion test to find the preconditions of splitting classes in the ontology refinement phase.

References

[1] Rocchio, J., Relevance Feedback in Information Retrieval, in Salton: *The SMART Retrieval System: Experiments in Automatic Document Processing*, Chapter 14, Prentice-Hall, pp. 313-323, 1971.

[2] Joachims, T., A Probabilistic Analysis of Rocchio Algorithm with TFIDF for Text Categorization, *Proc. of the 14th International Conference on Machine Learning (ICML97)*, pp. 143-151, 1997.

[3] Haav, H.M., Lubi, T.L., A Survey of Concept-based Information Retrieval Tools on the Web, *Proc. of the 5th East-European Conference ADBIS'2001*, Volume 2, pp. 29-41, 2001.

[4] Kuo, Y.H., Wong, M.H., Web Document Classification based on Hyperlinks and Document Semantics, *Proc. of PRICAI 2000 Workshop on Data Mining*, pp. 44-51, 2001.

[5] Ghasem Aghaie, N., Sarafan, G., Extraction of category and subcategories in an intelligent text categorization system, *Proc. of the Information and Knowledge technology conference (IKT)*, pp.446-454, 2003.

[6] Lu, C, Drew, M.S., Construction of a Hierarchical Classifier Schema using a Combination of Text-Based and Image-Based Approaches, *Proc. of ACM SIGIR 2001 The 24th Annual Conference on Research and Development in Information Retrieval*, pp. 438 - 439, 2001.

[7] Shamsfard M., Kazemeyni, F. & Khordad, M, Categorizing HTML Documents exploiting Ontology and Semantic Lexicon (in Persian), *Proc. of 10th Conf. of Computer society of Iran (CSICC'05)*, to be appeared, 2005.

[8] Joachims, T., Text categorization with Support Vector Machines: Learning with many relevant features, *Proc. of 10th European Conf. on Machine Learning (ECML)*, pp. 137-142, 1998.

[9] Gehrke, J., Rmakrishnan, R. & Ganti, V., Rain Forest-A Framework for Fast Decision Tree Construction of Large Data Sets, *Data mining & Knowledge Discovery*, Volume 4, pp. 127-162, 2000.

[10] Han, E.H., Karypis, G. & Kumar, V., Text Categorization Using Weight Adjusted K-nearest Neighbour Classification, *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 53-65, 2001.

[11] Ragas, H., H.A. Koster, C., Four Text Classification Algorithms Compared on Dutch Corpus, *Proc. of Annual ACM conf. SIGIR*, pp. 369 - 370, 1998.

[12] Yu, H., Han, J. & Chang, K.C.C., PEBL: Web Page Classification without Negative Examples, *Knowledge and Data Engineering*, **16(1)**, pp. 70-82, 2004.

[13] WordNet, http://www.cogsci.princeton.edu/~wn/.

[14] Lesk, M., Automatic sense disambiguation: How to tell a pine come from an ice cream cone, *Proc. of SIGDOC conference*, pp. 24 - 26, 1986.