Automatically Learning Document Taxonomies for Hierarchical Classification

Kunal Punera
Dept. of Electrical and Computer Engineering
University of Texas at Austin
kunal@ece.utexas.edu

Suju Rajan
Dept. of Electrical and Computer Engineering
University of Texas at Austin
suju@ece.utexas.edu

Joydeep Ghosh
Dept. of Electrical and Computer Engineering
University of Texas at Austin
ghosh@ece.utexas.edu

ABSTRACT

While several hierarchical classification methods have been applied to web content, such techniques invariably rely on a pre-defined taxonomy of documents. We propose a new technique that extracts a suitable hierarchical structure automatically from a corpus of labeled documents. We show that our technique groups similar classes closer together in the tree and discovers relationships among documents that are not encoded in the class labels. The learned taxonomy is then used along with binary SVMs for multi-class classification. We demonstrate the efficacy of our approach by testing it on the 20-Newsgroup dataset.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Content Analysis and Indexing; H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

Automatic taxonomy learning, Hierarchical classification

1. INTRODUCTION

Hierarchical classification algorithms have been applied to Web documents in order to alleviate the problem of maintaining and classifying documents in topic taxonomies. Prominent existing techniques [1][3] learn from a pre-defined taxonomy of documents. In this paper, we introduce a new approach that extracts the hierarchical structure automatically from a corpus of labeled documents. Furthermore, we show that our algorithm caters to the multi-modal nature of classes often found in Web data. Since such classes contain documents on diverse topics, they simultaneously belong in possibly very different parts of the taxonomy. Our algorithm identifies these classes and splits them, placing the subclasses in appropriate subtrees. Once the hierarchical structure is learned, we employ a binary SVM classifier [4] for each non-leaf node of the tree. Our technique groups similar classes closer to each other in the binary tree, ensuring that most classification errors are made among adjacent leaf nodes. We demonstrate on the 20-Newsgroup dataset that our algorithm learns meaningful taxonomies with no manual intervention.

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

2. APPROACH

Let $\{C_i\}_{i=1}^n$ denote the n classes in the given dataset and let $\{D_i\}_{i=1}^n$ indicate the corresponding sets of documents. We want to induce a binary tree T from this data such that each node in T corresponds to a set of classes; with the leaf nodes corresponding to single classes. Let $n(j)$ indicate the j^{th} node of T, so that $n(1)$ is the root, and $n(2*j)$ and $n(2*j+1)$ denote the left and right children of $n(j)$ respectively.

At the start of the algorithm we place all the classes at the root node. The given set of classes is first split into two sets of classes, and each such set is partitioned recursively until it contains only one class. The partitioning of a parent set of classes into two is described more formally below.

Initialize the root node of the tree T as $n(1) = \{C_i\}_{i=i}^n$. Repeat for each non-singleton node $n(j)$:

1. Select a set of discriminant features using the Fisher index criteria [1].
2. For each class at this node, calculate the mean document vector. Find the two classes whose mean vectors are farthest from each other. Initializing with these two centroids, cluster all the documents assigned to the node into 2 clusters using Spherical K-Means [2].
3. For all classes C_i assigned to node $n(j)$, let DC_{i1} and DC_{i2} be the set of documents clustered into the clusters 1 and 2 respectively.
   (a) If $|DC_{i1}| > (\theta * |D_i|)$ then assign class C_i and all its documents to the left child $n(2*j)$.
   (b) If $|DC_{i2}| > (\theta * |D_i|)$ then assign class C_i and all its documents to the right child $n(2*j+1)$.
   (c) Else create new sub-classes C_{i1} and C_{i2} from C_i such that C_{i1} contains documents DC_{i1} and C_{i2} contains DC_{i2}. Assign class C_{i1} to $n(2*j)$ and C_{i2} to $n(2*j+1)$.
4. Learn a SVM classifier to distinguish between the documents in the left and right child of node $n(j)$.

The process stops when all the leaf-nodes have only one class. Hence, we can now classify and move a test document down the tree until it reaches a leaf node. The class at the leaf node will be the label assigned to the document.

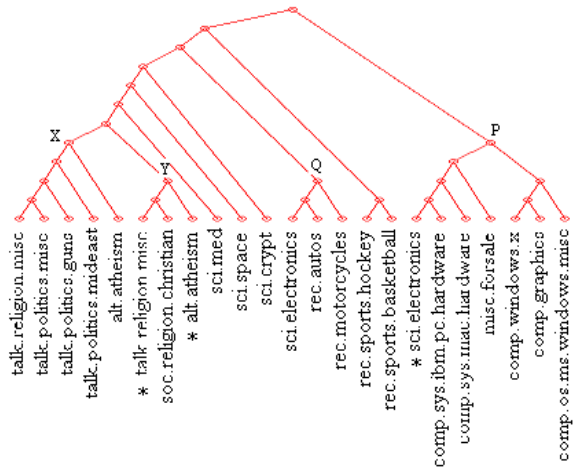

Figure 1: The tree structure generated from the 20-Newsgroup data

We assign a class (and all its documents) to a child node when more than a fraction (θ) of its documents go into any one cluster. If this is not the case, we break-up the class into two classes as described in step 3.*c* of the algorithm. Both parts of the split class point to the same class label. We allow classes to be split since we expect that some of them would contain documents on a diverse set of topics. There is, however, a danger of over-training since repeated partitioning of a class might lead to very specialized sub-classes. In order to prevent this we split subclasses only when they contain more than a β fraction of the documents of the original class they were created from. Both parameters θ and β can be tuned by using a validation set.

3. CASE STUDY WITH 20-NEWSGROUPS DATA

To demonstrate our approach, we evaluated it on the 20-Newsgroup dataset [5]. The dataset has 20 classes of roughly 1000 documents each. The data was preprocessed to remove headers, stop words, and words that occur less than 5 times leaving us with a vocabulary of 50736 words. 700 documents from each class were used for learning the tree. The validation set and the test set consisted of 100 and 200 documents respectively. For Spherical K-Means we reduced the dimensionality using the Fisher Index criterion and normalized the features by the IDF. Using the validation set we optimized over different numbers of features. The taxonomy in Figure 1 was obtained by setting $\theta = 0.7$, $\beta = 0.5$, Number of features=2000. Variations in parameter values caused only minor differences in tree structure and classification accuracy.

In Figure 1, leaf nodes are labeled with the corresponding classes. As can be seen in the figure, *Sci.Electronics*, *Alt.Atheism*, and *Talk.Religion.Misc* split up into two subclasses each (marked with asterisk). Table 1 lists the words that best discriminate between these subclasses. The documents in the *Talk.Religion.Misc* class have two major themes: Politics and Religion. Our algorithm splits this class into two parts placing one each under subtree X and Y in Figure 1. Subtree X contains the classes talk.polictics.*, while Subtree Y contains classes relating to Religion. The *Alt.Atheism* class also splits up along similar lines. The Sci.Electronics class is bimodal too, and splits up at the root level of the tree. One part containing documents on Computers and the Internet falls under subtree P, which also holds all the Comp.* classes. Similarly, the other part of Sci.Electronics class contains documents on Automobiles and falls under subtree Q. This validates our idea that splitting broad classes would lead to better placement of documents in the hierarchy.

Class-name	Split	Salient words
Sci.Electronics	35%	Police, Radar, Battery, Car
*Sci.Electronics	65%	Email, Software, Chip, Data
Talk.Religion	40%	FBI, Government, Values, Pay
*Talk.Religion	60%	God, Jesus, Faith, Bible
Alt.Atheism	40%	Objective, Morality, Moral
*Alt.Atheism	60%	God, Religion, Faith, Bible

Table 1: Words with high fisher index for the split classes

The SVMLight package [4] was used to train SVMs with linear kernels for the binary problems induced at each node of the hierarchy. The 'C parameter' was set as 0.1 by tuning using the validation dataset. SVMs were trained on the term frequencies of the documents. Feature selection was not performed since it did not result in any improvement in classification accuracy for this dataset. Previous experimental studies [5] have also shown similar findings in the context of the One-vs-All algorithm. Our approach's performance in terms of F measure over the test dataset was 0.76. We also tested a simpler version of our approach which built the hierarchy without splitting classes. This gave us a F measure of 0.73 over the test dataset.

4. CONCLUSION AND FUTURE WORK

We showed that the hierarchy reveals interesting relationships between the classes. We plan to investigate how this information can be used to obtain lower classification errors between related classes. Since hierarchical classification methods scale well with increases in data, we also intend to experiment with larger Web datasets.

Acknowledgments: This work was supported by NSF (Grants IIS-0307792 and IIS-0312471).

5. REFERENCES

[1] 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. *VLDB Journal: Very Large Data Bases*, 7(3):163–178, 1998.

[2] I. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In R. Grossman, G. Kamath, and R. Naburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.

[3] S. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR '00*, pages 256–263. 2000.

[4] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184, 1999.

[5] J. Rennie and R. Rifkin. Improving multiclass text classification with the support vector machine. In *MIT. AI Memo AIM-2001-026*, 2001.