Text Categorization through Multistrategy Learning and Visualization

Ali Hadjarian[1,2], Jerzy Bala[2], and Peter Pachowicz[1,2]

[1] Sigma Systems Research, Inc., 3975 University Dr., Suite 215,
Fairfax, Virginia 22032 U.S.A.
{ahadjarian, ppach}@sigma-sys.com
[2] School of Information Technology and Engineering, George Mason University,
Fairfax, Virginia 22030 U.S.A.
jbala@gmu.edu

Abstract. This paper introduces a multistrategy learning approach to the categorization of text documents. The approach benefits from two existing, and in our view complimentary, sets of categorization techniques: those based on Rocchio's algorithm and those belonging to the rule learning class of machine learning algorithms. Visualization is used for the presentation of the output of learning.

1 Introduction

The recent explosion of on-line information has generated an urgent need for more effective and more "intelligent" tools for information access. The enormous shift from centralized to distributed computing in recent years and the replacement of mainframes and stand-alone computers by Internet and Intranet environments has played a key role in the increase of available on-line information. The World Wide Web, for instance, has provided its ever growing community of users with a tool which allows them to not only access millions of on-line information sources, but also to make their own information sources available to millions of other users around the world.

Text Categorization (TC) or the automatic assignment of documents to predefined topics has been recognized as an effective tool for managing the huge number of on-line documents. TC gives organization to the often unorganized document collections and hence eases the task of locating relevant information for the user.

The aim of this paper is to introduce a multistrategy learning approach to TC. A close look at two popular algorithms used for TC: a modified version of Rocchio's algorithm [1] and the rule learning algorithm RIPPER [2] reveals that despite their differences, the two are somewhat complimentary in nature. Our hope is to take advantage of the complimentary nature of these two algorithms by combining them in a unique multistrategy framework. Visualization is used for the presentation of the output of learning and helps the user in determining the category (or categories) of an incoming document.

A. Gelbukh (Ed.): CICLing 2001, LNCS 2004, pp. 437-443, 2001.

In the remainder of the paper, we will briefly discuss the two algorithms involved, present the multistrategy framework, explain the visualization aspect of the methodology, and finally present our conclusions.

2 The Individual Learning Algorithms and Their Characteristics

In this section, we will discuss the two learning algorithms employed in the proposed multistrategy system and mention some of their characteristics.

2.1 RIPPER Rule Learning Algorithm

One of the algorithms employed in our approach is RIPPER [2]. RIPPER is a rule learning program that has proven to be an effective tool for building text classifiers [3]. The input to RIPPER consists of a set of training text documents that have already been classified into predefined categories. Its output is a text classifier in the form of classification rules. A single rule can be a single condition or a conjunction of two or more conditions. The ruleset itself is a disjunction of the single rules.

RIPPER's learning strategy is very close to that of a decision tree classifier. The RIPPER ruleset is constructed through two separate phases: the learning phase and the optimization phase. During the learning phase, the ruleset is constructed by starting with an empty ruleset and then adding one rule at a time until all the positive training examples are covered. The rules themselves are constructed in the following manner. The training examples are first split into growing and pruning sets. Each rule originally has no conditions. Conditions are repeatedly added to each rule until no negative training examples, in the growing set, are covered by that rule. The resulting rule is then simplified by removing one or more conditions so as to improve the rule's performance on the examples in the pruning set. The greedy search for new conditions to be added (during rule construction) or to be removed (during simplification) is done using ad hoc heuristic measures such as information gain [4].

During the optimization phase, the ruleset obtained in the learning phase is further optimized by reducing its size and improving its fit to the training examples. Each rule may potentially be replaced by an optimized rule that improves the overall performance of the whole ruleset. Rules are grown and simplified (like in the learning phase) so as to improve the performance of the whole ruleset on yet another held-out pruning set.

2.2 Rocchio's Algorithm

The second algorithm employed in the proposed approach is a version of Rocchio's algorithm [5], which has been modified for text categorization [1] and implemented by [3]. Given the vector space representation of individual documents (e.g. a vector containing the terms in the corpora and their corresponding weights for the given document), Rocchio calculates a prototype vector for each document category. This prototype vector has the same dimension as the original weight vectors. The weight of a

given term in the prototype is a combination of its weight in the relevant (i.e. belonging to the target category) and non-relevant (i.e. not belonging to the target category) documents and is calculated using the following formula:

$$\omega_{c,k} = \max\left\{0, \beta \cdot \frac{\sum_{i \in relevant} \omega_{i,k}}{n_{relevant}} - \gamma \cdot \frac{\sum_{i \in non-relevant} \omega_{i,k}}{n_{non-relevant}}\right\} \quad (1)$$

where $\omega_{c,k}$ is the weight of the k^{th} term in the prototype vector for category c, $\omega_{i,k}$ is the weight of the k^{th} term in the vector representation of document i, *relevant* is the set of all documents belonging to category c, *non-relevant* is the set of all documents not belonging to category c, $n_{relevant}$ is the number of documents belonging to category c, and $n_{non-relevant}$ is the number of documents not belonging to category c. β and γ are parameters which control the contributions of relevant and non-relevant documents (i.e. positive and negative examples) to the prototype vector, respectively. The standard values for these parameters are $\beta = 16$ and $\gamma = 4$ [6].

There are many different ways to calculate the weight vector for any given document. The weight vector for document i is represented as $\langle \omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \ldots, \omega_{i,t} \rangle$, where t is the total number of indexing terms. One of the most popular weighting schemes is the TF-IDF (Term Frequency – Inverse Document Frequency) approach, which is based on the notion that a word with high frequency within a document is perhaps representative of the document content and so it should be given a high weight and a word appearing in too many documents is perhaps not very discriminatory and so it should be given a low weight [7].

Using Rocchio, a document is classified as belonging to category c, if its distance (i.e. dot product) to the prototype vector for category c is less than some threshold. The threshold can, for example, be set to a value that minimizes the classification error rate on the training data.

2.3 RIPPER vs. Rocchio

Here we will discuss some of the characteristics of the two categorization techniques described above and by highlighting their differences, which make the two somewhat complimentary in nature, we hope to justify their integration within a unique multistrategy framework.

Rocchio, being a linear classifier, generates just a single model for each category of documents in the form of a prototype vector in a vector space model. RIPPER, on the other hand, has the ability to generate multiple models for the same document category. In other words, every rule generated by RIPPER can potentially be considered as a model that tries to capture some characteristics of the training data. This is especially important if the documents belonging to a single category form disjoint clusters in the representation space (e.g. Science category containing documents on Astronomy, Biology, etc.).

RIPPER generates classification rules that are easily comprehensible by people. Rocchio generates category prototypes in the form of vectors of numerical values (i.e. term weights) that are perhaps not as intuitive.

RIPPER's classification rules are discriminatory in nature. In other words, the rules include only those conditions (e.g. words) that are sufficient for discriminating among the various training categories and hence are often too general. The prototype vectors produced by Rocchio, on the other hand, are more descriptive. This means that a number of relevant words may contribute to the categorization task, not just those few that are highly discriminatory.

RIPPER allows for the inclusion of *context* into the classification process. In other words, the influence of any given word on classification is sensitive to the presence or absence of other words in the document. Moreover, RIPPER itself determines the context. Rocchio on the other hand is not context-sensitive. In other words, it assumes that the influence of any given word on classification is independent of the presence or absence of any other words in the document.

A major advantage of Rocchio is that it provides a measure of similarity or "typicality" for a test document. In other words, using Rocchio, we are able to tell how close a given test document is to the learned category prototypes. This is achieved through the calculation of some kind of distance (e.g. dot product) from the vector representation of the test document to the vector representation of the prototype vector for a given category. The similarity measure can be viewed as a classification confidence factor. RIPPER, on the other hand, does not have the ability to assign a confidence measure to its classification decisions. In other words, it classifies a given test document as belonging to one of the existing categories without giving an indication of its "typicality" (except for using ad hoc measures such as rule strength). This disadvantage is common among rule learners.

3 A Multistrategy Approach

The proposed approach tries to take advantage of the complimentary nature of the above algorithms by combining them in a multistrategy framework. More specifically, the approach generates multiple models of categories in the form of discriminatory decision rules that are easily comprehensible by people (i.e. First learning phase). But instead of categorizing an incoming document by simply applying these rules, the algorithm uses more descriptive models of the categories that can also be used to measure the typicality of a document (i.e. Second learning phase). Here we will describe the architecture in more detail.

The overall view of the approach is depicted in Figure 1. The input to the learning system consists of a number of training documents annotated by their topic. Given the initial categories of documents and example documents belonging to each category, the system will try to generate descriptions of these categories by using the RIPPER rule learning algorithm. The resulted output of this first phase of learning is a set of discriminatory rules for each of the training categories (e.g., two rules for Topic1 and three rules for Topic2 in Figure 1).

A nice by-product of the rule learning process is an implicit clustering of the training documents. In other words, those documents classified by a single rule could form their own cluster within the representation space. The common characteristic of the documents in a single cluster is of course the presence of a term (or terms) as indicated by the conditions of the classification rule. As indicated earlier, the documents belonging to a single category may form disjoint clusters in the representation space. Different rules in the learned ruleset try to cover these disjoint areas. Unlike a rule learning algorithm such as RIPPER, a linear classifier such as Rocchio would generate a single model at the center of all the disjoint clusters without accurately representing each single cluster. This is why algorithms such as RIPPER are much more suited for learning concepts formed by disjoint document clusters.

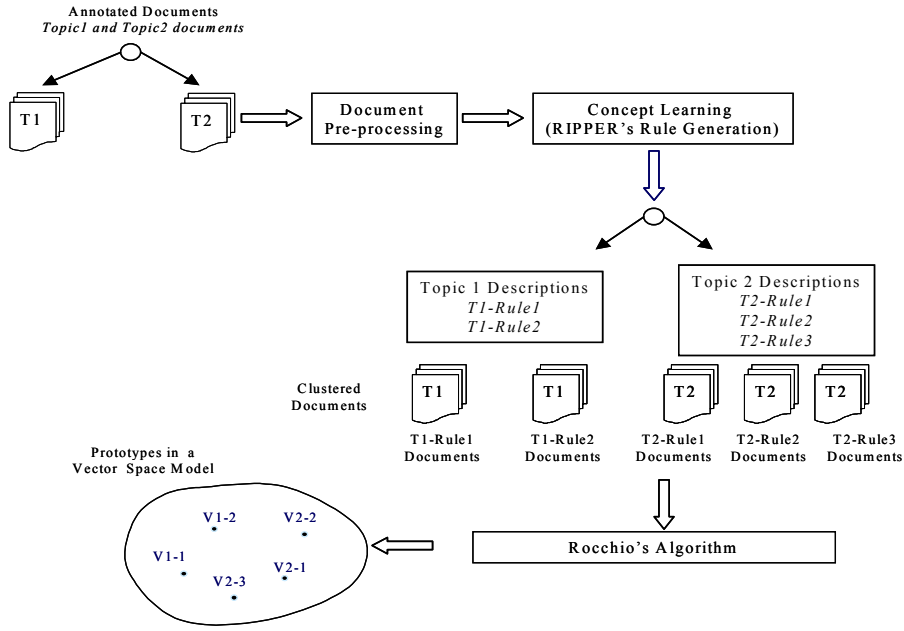

Fig. 1. A multistrategy approach to document categorization

The function of the second learning phase (i.e. Rocchio) is to generate more descriptive rules for each topic. The advantage of having more descriptive rules is that in the absence of a very large and comprehensive training set, the discriminatory rules learned during the first learning phase may tend to be too general. A more descriptive model requires the presence of a number of relevant terms in order to correctly categorize a given document and not just a few highly discriminatory terms.

Here, however, instead of learning a single prototype for each category of documents as it is traditionally done, the system learns a different prototype for each rule generated in the first phase. As described earlier, a nice by-product of rule learning is the implicit sub-clustering of documents in a given category. In other words, all the documents that are "explained" by a given rule are clustered together. Rocchio gener-

ates a different prototype vector for each cluster of documents (i.e. instead of generating a single prototype at the center of disjoint document clusters, which may not be very accurate).

Now when trying to match an incoming document, instead of directly applying the rules obtained during the first learning phase, the system calculates the "distance" of this document to the prototype vectors. And by presenting the results visually (i.e. as explained in the following section), the system can assist the user in identifying the potential category (or categories) of an incoming document.

4 Visualization of the Output

Figure 2 depicts a sample output of the multistrategy categorization system described above. The document categories used for this particular example come from the popular Reuters-21578 (Distribution 1.0) dataset. This dataset along with its detailed description is available through David D. Lewis's homepage [8]. Here four of the categories (e.g. "Grain", "Trade", "Shipping", and "Crude oil") were chosen for the categorization task at hand. As it can be seen in the figure, there are a number of boxes (i.e. 3D bars) associated with each category. Each box represents a cluster of documents formed by a single rule generated for that category. The height of the box represents the number of documents in the cluster. The key words associated with each particular cluster (i.e. the conditions of RIPPER's rules) can be seen by brushing the box. For example the brushing on the box on the lower portion of the figure reveals that the two most important key words for that particular cluster of documents belonging to the "Crude oil" category are OIL and BARRELS.

The color of each box determines the degree of similarity of an incoming document to each cluster (as determined by the second learning phase: Rocchio). Here for example a darker color indicates a stronger match and so we can infer from the figure that the incoming document belongs to the "Crude oil" and to a lesser degree to the "Shipping" categories.

5 Conclusions

The multistrategy approach presented here combines the benefits of a rule learning algorithm, such as compact and easily comprehensible representation of the learned knowledge and the ability to generate multiple models of the same category, with those of the more traditional Rocchio's algorithm, such as the generation of more descriptive models of categories and providing the means for measuring the closeness of a given document to each category.

This is an ongoing research project. Aside from the immediate benefits highlighted above, extensive sets of experiments are needed to measure the gain, if any, in the overall classification accuracy by combining the individual learning strategies. Future work also includes the implementation of more elaborate modeling techniques for identifying the disjoint clusters of documents belonging to the same category. Docu-

ments represented by different rules may not necessarily belong to disjoint clusters as it has been assumed here.

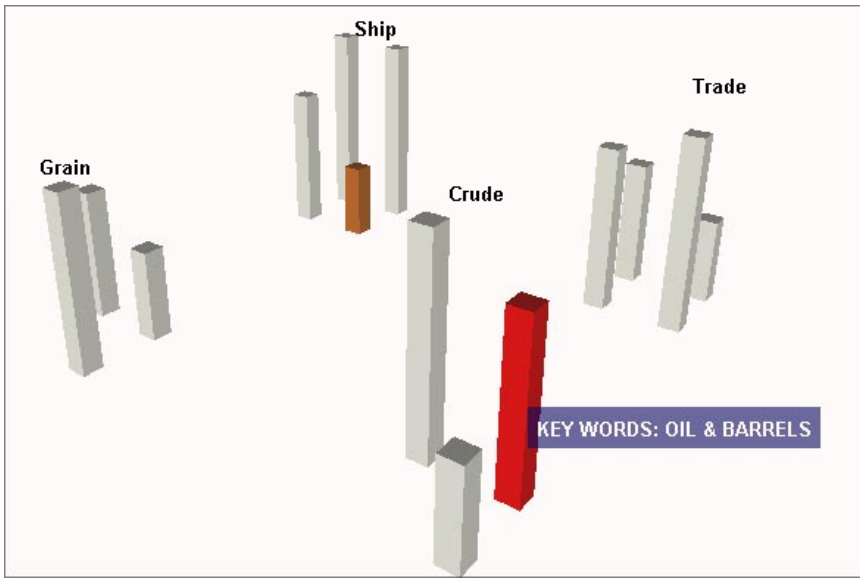

Fig. 2. Sample output

References

1. Ittner, D. J., Lewis, D. D., and Ahn, D. D.: Text categorization of low quality images. In Symposium on Document Analysis and Information Retrieval. Las Vegas, NV. (1995) 301-315
2. Cohen, W. W.: Fast effective rule induction. In Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, California. (1995)
3. Cohen, W. W. and Singer, Y.: Context-sensitive learning methods for text categorization. In ACM Transactions on Information Systems 17, 2. (1999) 141-173
4. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA. (1993)
5. Rocchio, J.: Relevance feedback information retrieval. In G. Salton, ed. The Smart retrieval system: experiments in automatic document processing. Prentice-Hall, Englewood Cliffs, NJ. (1971) 313-323.
6. Buckley, C., Salton, G., and Allan, J.: The effect of adding relevance information in a relevance feedback environment. In Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland. (1994) 292-300
7. Salton, G.: Developments in automatic text retrieval. Science 253. (1991) 974-980
8. Lewis, David D.: Homepage available at http://www.research.att.com/~lewis.