CS224N: Final Programming Project

Three Pandas: Laura Back & Alexei Kosut

June 8, 2001

Abstract

This project explores the problem of categorizing documents into predetermined groups, based on rules learned from a training set of categorized documents. Our goal was to explore two ways that keyword recognition can be used to define rules: by associating the presence of a particular keyword with a specific category, and by associating a high density of a keyword with a category. Our work explored the questions of whether one of these models performed better than the other and of whether a hybrid model could achieve the best performance of all.

1 Introduction

The problem of document categorization involves teaching a machine to sort texts into a predetermined set of categories based on their content. The categories and texts should be such that a human reader who understands the subect matter can easily do the categorization. Most work in the field, rather than attempting to instill some high-level understanding of content in the machine, looks for heuristics that allow the categorization to be done based on simpler measures. One of the simplest measures is keyword recognition–associating words with particular categories based on information extracted from a training set of documents, and then using the words in a document to be categorized to decide which set of training documents it best fits into.

Much of the work in keyword-based document categorization focuses on developing statistical models that represent the information found in the training set and provide a framework for applying information found in the unknown document. Our research took a different angle, looking not at the statistical model for handling information but at the nature of the extracted information itself. A simple type of information to glean from keywords is binary–whether a given word tends to be present in a particular category of documents. A slightly more complicated approach uses measures of word frequency within documents, matching the frequency of a word in an unknown document with its frequency in documents of each category. Our project compares the simple binary approach with a frequency-based

approach. The frequency-based approach we use is a simplified one, recognizing words that tend to have particularly high frequency in a certain category, and choosing that category when those words have high frequency in the unknown document. (We do not test an approach which tracks the typical frequency of a word in each category and then attempts to find the closest match, rather than just matching high frequencies with each other.)

We compared these approaches with a program that tests the binary (presence-based) approach against the frequency-based approach by learning rules based on one of the above. In the first case, rules say "If word x appears in a document, that indicates that the document has category y." In the second, they say "If word x appears with high frequency in a document, that document has category y." We also implemented an attempt to combine these two types of rules by making rules of the first type for some words and of the second type for others, as appropriate. Our hope was that the hybrid implementation would prove more effective than the implementations that used a single strategy.

2 Motivation

This work was motivated by a desire to take a new approach to keyword-based categorization, exploring the nature of the keyword usage itself. We considered the simple presence-based approach and wanted to explore other information that might be gleaned from keywords to help categorize a document.

After a presence-based measure, frequency occurred to us as another obvious property of keywords that can be quantified, so we wondered if frequency measures could play any role in document categorization. Intuitively, it seemed as though it should, and we thought of several anecdotal cases where it might. In the case of ambiguous word sense, for instance, two meanings of a word might appear in entirely different categories of documents. It seemed likely to us that different senses, each occurring where it was relevant, would tend to occur with different frequencies.

As an example, take the classic "bank": a financial institution, or the side of a river. A financial article may make heavy use of the term, whereas it seemed feasible to us that a nature article might mention the bank of a river, but would be unlikely to focus the article around the bank and therefore unlikely to use the term repeatedly. Other words that seemed likely to have such properties would be those that had both a topic-specific and a non-topic related meaning: for instance, the word "general" in a military sense, and its generic use elsewhere. It seemed unlikely that the generic use would ever occur with high frequency, while the military use might in an article on a relevant topic.

We chose the most simple form of frequency checker (high vs. low) rather than a more complicated analysis because we believed that the simple check should be sufficient to show us whether frequency measures were in fact useful; a more complicated model could then be implemented in an attempt to improve on the original results.

3 Our algorithm

Since our goal was to test the type of information extracted, rather than the statistical model used to handle the information, we tested all approaches using a simplistic statistical model chosen primarily for its straightforward implementation. We assume that a more intelligent model would give improved performance for all approaches.

To make the rules that we used to categorize documents, we simply calculated, from the training set, the probability of a word appearing in documents of a given type and its average frequency when it did appear. From this we extracted rules as follows:

(i) if a word appeared with high probability in documents of a certain category and low probability in documents of other categories, we made a presence-based rule.

(ii) if a word appeared with at least moderate probability in documents of a certain category and had high average frequency when it appeared in that category, and low average frequency when it appeared in documents of other categories, we made a frequency-based rule.

(iii) in the hybrid implementation, if a word was a candidate for both types of rule, we made a presence-based rule.

The reasoning behind (i) should be straightforward, but (ii) and (iii) required a few design decisions that merit explanation.

First, in making frequency-based rules, we required that the word appeared with at least moderate probability in documents of this category. We expected that a word that appeared only in a few documents would not give us enough data to make any assumption about its typical frequency in that category, so any rule created would have been as likely as not to give misleading information.

Second, in tracking frequency information, we calculated average frequency of a word in a category only over the documents in that category in which the word appeared. We applied rules to an unknown document based on the words that appeared in that document, so once a word was found in the unknown document, we believed the measure of expected frequency given the word's appearance to be more relevant to the unknown document than an expected frequency that ignored this information. Additionally, incorporating zeroes into the average frequency information would have introduced a presence-based component to the measure, and we wanted to keep it strictly frequency-based.

Finally, in the hybrid implementation, we gave presence-based rules precedence over frequency-based ones. Since a word that qualified for a precedence-based rule by definition appeared with low probability in documents of other categories, a presence-based rule seemed more valid: the frequency information we had for the other categories, being based only on a very few documents, was not expected to be statistically significant, reducing the significance of the frequency comparison.

We also gave a weight to each rule. For presence-based rules, the weight was the fraction of

training documents in which the word appeared (words that had the highest probabilities of occurring were taken as the strongest indicators), and for frequency-based rules, the weight was the word's average frequency where it appeared (again assuming that words with higher frequencies gave more decisive rules).

Once training rules were formulated, we categorized documents by collecting keyword information and, for each word in the document, seeing if a rule existed for that word. If the word in the given document fit the criteria for the rule's applicability (no criteria beyond presence for presence-based rules; a minimum density for density-based rules), the rule's weight (multiplied by a weight for the type of rule in the hybrid implementation) was added to a counter for that category. After all possible rules were applied, we chose the category that had accumulated the highest weight for that document.

The weight for type of rule mentioned above was based on whether the rule was presence-based or frequency-based. Since rule weights themselves were determined in a way that was dependent on the type of rule, we expected some kind of normalization to be necessary to combine the types in appropriate proportion. We set the parameter values experimentally.

4 Parameters

There were eight parameters that we tuned to try to optimize performance. These were:

- The minimum fraction of training documents in a category a word had to appear in to be considered for a presence-based rule (HIGH_PROB).
- The maximum fraction of training documents a word could appear in in any category to be considered for a presence-based rule in some other category (LOW_PROB).
- The minimum average frequency a word had to have in training documents where it appeared in a category before it could be considered for a frequency-based rule in that category (HIGH_DENSITY).
- The maximum average frequency a word could have in any any category to be considered for a frequency-based rule in any other category (LOW_DENSITY).
- The minimum fraction of training documents in a category a word had to appear in to be considered for a frequency-based rule, as discussed above (MID_PROB).
- The frequency a word had to have in a testing document before a frequency-based rule could be applied (DENSITY_THRESHOLD).
- The weight given to presence-based rules (BINARY_WEIGHT).
- The weight given to frequency-based rules (DENSITY_WEIGHT).

All of these parameters were tuned experimentally to try to optimize performance. The precise optimal values are unlikely to be interesting, since the precise measure is likely to be more a function of the training and testing data than of any inherent properties of the measures; however, we did see some noteworthy general trends.

5 Training Data

To train and test our categorizer, we used a subset the Reuters-21578 corpus. Our primary reason for choosing this corpus was that it was the largest body of text available to us that was tagged for topic in a reasonable way. Also important is that this corpus is treated as a benchmark corpus by other researchers in the field of text categorization; most of the papers we examined showed data based on tests against this corpus or an earlier version, Reuters-22173. Since we are examining a different aspect of the text categorization problem then other research, we did not compare our results directly, but using the same corpus means that our results are applicable to the same problem as other research.

Rather than use the entire Reuters-21578 corpus, we elected to train on a smaller subset, with fewer categories and fewer documents. This was done to both simplify the problem and give us more time to test our algorithm's parameters, and so we could be sure to choose categories that were meaningful and for which we had a large set of training data. We categorized the Reuters-21578 corpus based on topic, using the Modified Apte split. We then chose the ten categories with the most documents (98 or more, excluding the ones marked "brief"). We then chose 98 from each of those categories, and used the resultant 980 documents to as our training set. We did so that our algorithm would generate data equally from all categories, and so we did not have to be concerned with weighting our rules appropriate to the category's size. After training, we tested our categorizer on the 2322 documents selected by the split as training documents in our ten categories.

To format the documents for use with our categorizer, we wrote a Perl script (named format in the submitted code) which read in the Reuters-21578 SGML data, seperated the articles into seperate files, stripped them of SGML tags and entities, categorized them into topics, and generated directories and files of testing and training data appropriate to our program. This winnowing gave us 4.6 megabytes of total data to work with, down from the 25 megabyte Reuters-21578 whole.

6 Results

Because of the simplified statistical model we used, we did not expect any of our implementations to perform particularly spectaculary relative to other implementations that have been devised; rather, our interest was in comparing them to one another to see how the different types of rules performed. Our expectation was that both would perform moderately

well individually, and that the hybrid implementation would show a substantial performance increase over either.

When we set the parameters so that no rules were generated, we got 44% accuracy in our testing set merely by letting all documents default to a single category (which happened to have high frequency in our testing set).

Testing only presence-based rules, we were able to achieve maximum correctness values just above 80%. One interesting point here was the optimal tuning of the HIGH_PROB and LOW_PROB parameters: performance was better the closer together we set the two. We had expected that we would want some space between them–a space in which a word occurred with too high probability in a category to justify making a rule for some other category, but not high enough to merit a rule of its own. This space seemed not to exist. This may in part be because of the way we weight our rules: any rule based on a word occurring in such a space would have low weight relative to other rules if we made it, anyway, so it would be no more likely to do harm than good. Tuning to the nearest hundredth (greater precision being meaningless with only 98 training documents in a category), the value we found to be optimal for both parameters was quite low: 0.04. In the general case, this particular value is likely to be a function of the variety of content within a single category in the given data.

Almost regardless of parameter values, the performance for the frequency-based measures was abysmal: it peaked at 54%, only 10% better than the default. We found that we were not making many frequency-based rules no matter where we set the parameters. We generated the most rules when we set the parameters so that the frequency rules were essentially presence rules (a high density was any occurrence of the word; a low density was none), and this seemed to give the best performance, but it was still much lower than the actual presence-based measures. We can think of two possible reasons for this.

First, our ability to imitate presence-based rules as frequency-based rules was limited. There was no way to specify an allowable occasional presence of a word in a document of another category (anything below the LOW_PROB parameter value described above); the only approximation was setting the low frequency threshold at zero, equivalent to a low probability threshold of zero. Similarly, there was no way to specify a specific high probability, only to set the maximum frequency threshold high enough to admit the presence of any word (and thus generate a rule with a HIGH_PROB value that allowed a word to appear in only one document). We therefore could imitate a presence-based rule engine, but not a very good one; we were generating bad rules as well as omitting good ones. Using similar values in the presence-based engine also gave degraded performance, although not quite so badly.

The additional degradation can probably be blamed on the fact that our weights for rules were inconsistent with what they would have been in a presence-based engine. If this is the case, it suggests that at least our scheme for weighting presence-based rules turned out to be somewhat useful; it may be our one positive result.

Apart from trying to make it imitate the presence-based checker, we could do very little with the parameters we had to make the frequency-based checker work better; we were able to get our maximum performance by playing with the parameters HIGH_DENSITY,

LOW_DENSITY, and MID_PROB, but the sensitivity of the system's performance to any of these was very slight.

Our hybrid implementation fared little better; we could not gain more than a percent improvement over the presence-based checker.

7 Related Work

There have been a number of active research topics in text categorization over the past few years, and prior to embarking on our project, we examined several papers related to the topic. Most of the research work in text categorization has focused on applying and improving statistical and machine-learning methods to classify text based on words from a training set. These approaches include Support Vector Machines, k-Nearest Neighbor clasifiers, neural networks, Linear Least-squares Fit mappings, Naive Bayes classifiers, and others.

While our implementation idea is also statistical in nature, it focuses on a different part of the problem than most of these ideas, and we were not interested in the problem of statistically modeling the text categorization problem. In fact, the statistical model we used was rather simple and ineffectual compared with any of these, but allowed us to test directly the effectiveness of our research, and we expect that our results would be similar regardless of statistical model.

The closests related work we found was by Yang and Wilbur in applying Wilbur-Sirotkin stop word identification to text categorization. Their study, "Using Corpus Statistics to Remove Redundant Words in Text Categorization," studies using a metric of "word strength" to identify the words most important to a particular category. Their method then identified unimportant words and removed them from consideration from the categorization model. This is similar to what we tried with our models; our density rules were generated based on words that seemed important to one category but unimportant to others. This is an improvement upon the stop-word idea, which does not take into account the relative differences between categories, only the words the algorithm finds important to the category itself. We did not directly ignore unimportant words, but since our algorithm only generated rules for words that passed our importance thresholds, a similar effect was realized.

8 Conclusions

In light of our results, it seems clear that our attempts to use frequency-based keyword information to improve document categorization were wholly unsuccessful. It is not impossible that there actually is useful information to be gleaned from word frequencies. The rough high/low frequency distinction may not be sufficient to use this information in any meaningful way. Alternatively, perhaps there is some value to the measure, and our particular

implementation was merely so badly done as to manage to avoid assessing it correctly. Although there may be merit in keyword frequency-based methods to improve categorization, our research did not provide results to support this.

9 Bibliography

Manning, Chris and Hinrich Schutze. *Foundations of Statistical Natural Language Processing.* pp 575–608.

Yang, Yiming and Xin Liu, "A re-examination of text categorization methods". *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGR), 1999, pp 42–49.

Yang, Yiming, "An Evaluation of statistical approaches to text categorization." *Journal of Information Retrieval*, 1999, Vol 1, No. 1/2, pp 67–88.

Yang, Yiming and John Wilbur, "Using corpus statistics to remove redundant words in text categorization." *Journal of the American Society for Information Science* (JASIS), 1996.

Appendix: README

```
Laura Back (leback@stanford.edu)
Alexei Kosut (akosut@stanford.edu)

2 days late  (NOTE:  We received permission from Prof. Manning to pool
late days--Laura had five and Alexei had none, so Laura is giving two of
hers to Alexei.)

Files submitted:

*.cc, *.h:  Our source
Makefile, mkdep:  Used to build the project
writeup.pdf:  Writeup of our work
eval:  A Perl script to compare the results output by the program to the
  correct results (usage discussed below)
format:  The Perl script used to format the data for testing and training
  files
data/testfile:  The list of files in the testing set
data/trainfile:  The list of files in the training set
data/answerfile:  The list of correct answers for the testing set (same
  order as the files appear in testfile)
```

```
data/train/*:  The files in the training set
data/test/*  The files in the testing set


Usage of our executable:

final <trainfile> <testfile> <outfile> <num categories>
  <training per category> <implementation number>

<trainfile> is a file containing the a list of files in the training set
  (one per line).  They must be sorted by category, with an equal number
  in each category.

<testfile> is the list of files in the testing set, formatted as above.

<outfile> is the file to print output to.  it prints a category for each
  testing file, with a measure of certainty (the weight given to the
  selected category divided by the total weight of applied rules--see
  writeup for details on weight, categories, rules, etc.)

<num categories> is the integer number of categories to divide data into

<training per category> is the number of training documents per category

<implementation number> is 1 to run the presence-based rule
  implementation, 2 to run the frequency-based rule implementation.  omit
  this argument to run the hybrid implementation.

To run with our input (from the data subdirectory):

../final trainfile testfile outfile 10 98

(optionally followed by '1' or '2')


Usage of the eval script:

eval <outfile> <answerfile>

<outfile> is the file output by our program

<answerfile> is a list of correct answers; a single number (starting at
  zero, categories ordered as in the training file) on each line, ordered
  as in the testing file
```

Group member responsibilities:

Laura designed the implementation and wrote the code (except hashtable.h, which Alexei wrote--Laura hates templates) as well as doing testing and parameter-tuning. Alexei wrote the Makefile, researched related work and selected and organized the documents we tested our program on (including reformatting them to the program's specifications). The writeup was written jointly, with each of us primarily responsible for the sections relating to the part of the work we did. Alexei did the pretty formatting of the writeup. Laura wrote the README.