CSE473 Project #2

Text Classification

Due December 9, 2002 at 10:30 AM

Overview

The second project is to implement text classification algorithms and apply them to a standard data set, the Reuters collection of news stories. The goal is to predict the topic of each news story in the test set, given the topics of the stories in the training set. You will implement the following algorithms:

1. Naive bayes
2. Decision trees
3. Decision trees with bagging

Naive Bayes should be implemented as described in class and in Section 6.10 of Mitchell. Be sure to implement Naive Bayes as a sum of logarithms of probabilities, rather than a product of probabilities, to avoid numeric underflow problems.

When learning decision trees, each feature should be the presence or absence of a given word in the document (e.g., if the word "silver" appears in the document, the corresponding feature should be 1 for that document, otherwise it should be 0). As an extension, you may try making each feature the frequency of the word in the document (i.e., the number of times it appears in the document, over the total number of words in the document). This will be slower, but may give better results.

Decision trees should be implemented with chi-square pre-pruning (described on page 543 of Russell and Norvig) which you should figure out how to extend to multiple classes. Use 5% as the significance threshold.

For efficiency, use a relatively low number of replicates with bagging (try 10, 20, and 50 as time allows). If necessary, you can try having a hard limit on the number of nodes in each replicate decision tree (e.g. 100).

For all algorithms, you should pre-select the k features (words) with the highest information gain with respect to the class variable. Try out several values of k to see the tradeoff between accuracy and efficiency; we suggest you start with $k = 1000$.

You must turn in a hardcopy of your code with a short writeup described below. We may ask you for a demo and/or oral presentation of the project. You program should be written in C, C++, Java, C#, or Lisp.

Writeup

The writeup should be no more than four pages. Include a description of your architecture, approach, extra features, bugs, and any problems you had.

Include a table of results with test set accuracies for all algorithms and main variations tried. Comment on the results. Which algorithm(s) did best? Why? When does each algorithm tend to do well versus poorly? How would you further improve your best text classifier?

In addition, make clear which parts were done by which group member.

Files

Format

The format of the test and training files is the same. Each file contains a set of examples delimited by two blank lines. Each example is composed of the following parts.

1. topic classification
2. blank line
3. story title
4. blank line
5. story location, date
6. blank line
7. story text

URLs

This is the training data:

http://www.cs.washington.edu/education/courses/cse473/CurrentQtr/projects/training.dat

This is the test data:

http://www.cs.washington.edu/education/courses/cse473/CurrentQtr/projects/test.dat

These are the topics used for classifying the texts:

http://www.cs.washington.edu/education/courses/cse473/CurrentQtr/projects/topics.dat

The original source for this data is the UCI Knowledge Discovery in Databases Archive:

http://kdd.ics.uci.edu/

The Reuters-21578 Text Categorization Collection was the collection used to generate the test and training data above. The SGML was converted to XML, cleaned up, and queried using XPath to generate a simple text file format:

http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html