Text Categorization
Machine Learning & Data Mining
Benjamin Cheung

Introduction

Text categorization involves a process of grouping files into different classes. In defining text categorization in this project, it is assumed that the number of classes is known and fixed with each file assigned to exactly one of them. This process is a supervised learning problem as commonly addressed in machine learning. With online information increasing at a rapid rate, it is worthwhile to have an automated text categorization. The dataset "20 newsgroups" used for this project is made widely available. Further, a pre-processed dataset "20-newsgroups-bydate" is also available. The goal of the project was to implement a version of the Naive Bayes classifier. In addition of this, to increase performance, feature selection was implemented to help discriminate between classes. To extend the project further, due to the number of free libraries available for machine learning the project sought to compare performance across our own implementation of Naive Bayes to various supervised learning approaches within the "scikit-learn" library in Python.

Implementation

A sample of 18846 documents was obtained from the CSE website at http://qwone.com/~jason/20Newsgroups/ under "20news-bydate". These documents were pre-processed by date into 60% training and 40% test set with duplicates and some headers removed. Using python as my choice of programming language, I first began by pre-processing the dataset further by only including the body of text. The vocabulary was then built by tokenizing words through removing punctuation and making them lower case. A combination of three feature selection methods were utilized paralleled to Joachims, 1996: (1) Vocabulary pruning of infrequent words (2) Vocabulary pruning of high frequency words (3) Increasing probabilities of words with high mutual information with the target concept. In extension of the feature selection method (3), to help discriminate between classes and reduce noise for high common words with high information gain (IG) with target value, if these words had correlations with other target values (e.g. god in atheism, religion) a simple average was taken across the common word's IG and therefore IG for the class word was made constant between them (3E). The variables for feature selection was words occurring at least 2 times for (1), removing 100 most common words in (2), and simple averaging the common words in 5000 most frequent words in each class(3E). Following this, the Naive Bayes calculations were made in accordance with the approach in Table 6.2 (Mitchell, 1997).

Method

A sample of 19997 documents was obtained from http://qwone.com/~jason/20Newsgroups/ under "20news-19997". A python script was created to convert specific files into UTF format so the library could read it, and following this a Monte Carlo cross-validation method was used. In this method, the dataset was randomly split into 60% training and 40% test set 10 times. Four supervised learning approaches were investigated in this project being Naive Bayes, Logistic Regression, Support Vector Machine and Random Forest. The Multinomial Naive Bayes in the "scikit-learn" library has several variations being with term frequency (TF) and extra modification of inverse document frequency (TFIDF).

Results

Throughout results, the performance measure F1 Score will be used as it is widely used in document classification and can be used to provide a consistent measure between my implemented Naive Bayes to the "scikit-learn" library performance measure. The results from the implemented Naive Bayes on the pre-processed dataset "20news-bydate" are shown in Table 1 along with performance improvements following feature selection. The supervised learning approaches were also compared on the same dataset with results shown in Table 2. For consistency of results, a Monte Carlo cross-validation averaged F1 score over 10 splits are shown on Table 3.

Discussion

Seen from the results in Table 1, having a vocabulary pruning method of infrequent words (1) and high frequency words (2) improves the performance measure for Naive Bayes. However, including only high mutual information as a feature selection decreased the performance by itself, but could provide a maximum performance increase in conjunction with (1) and (2). As seen in table 2, our implementation in accordance with Table 6.2 (Mitchell, 1997) provided greater performance than the multinomial Naive Bayes without the inverse document frequency suggesting that a manual implementation may be better than a library in certain cases (Note: Since it is implemented manually, there may be some bugs but I am assuming it is correct). In the 20-newsgroup dataset sorted by date the maximum performance for implemented Naive Bayes was 83% with Linear Support Vector Classifier (SVC) as the best classifier at 85%. In the same dataset but using Monte Carlo cross-validation, as seen in table 3 the maximum performance for implemented Naive Bayes was 89% with Linear SVC at 92%. Explanations for why performance was lower in sorted date dataset may be due to future text in categories changing over time in emails (e.g. classification performance would be worse in using test set emails in 10 years compared to 1 year).

Related work

The approach taken to my own implementation of Naive Bayes is adopted from Mitchell 1997 book. The 3 text feature selection methods were also adopted from Joachims (1996). To the best of my knowledge no-one has implemented my extension of text feature selection by reduction of noise by simple average and is original through understanding of the content. Application of supervised learning approaches is widely used in text analytics.

Conclusions

In summary of results, it can be concluded that the feature selection increases performance in Naive Bayes, however a linear SVC is still supported widely as the best text classification algorithm. In ranking text classification methods from worse to best performance: Random Forest, Naive Bayes(TFIFD), Logistic Regression, Naive Bayes (Feature Selection) and then Support Vector Machine.

References

Mitchell, T. "Machine Learning". McGraw Hill, 1997.

Joachims, T. (1996). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, (Computer Science Technical Report CMU-CS-96-118). Carnegie Mellon University.

Appendix

Table 1
Performance Measure: F1 Score
"20news-bydate" dataset
Naive Bayes Implementation with Feature Selection

Feature Selection	**F1 Score**
Original	74.4956%
(1)	79.1697%
(2)	76.1860%
(3)	73.8693%
(3E)	68.1162%
(1),(2)	82.1098%
(1),(3)	78.5979%
(1),(3E)	80.4645%
(2),(3)	74.8839%
(2),(3E)	70.0373%
(1),(2),(3)	81.8585%
(1),(2),(3E)	83.0367%

(1): Vocabulary pruning of infrequent words less than 2
(2): Vocabulary pruning of high frequency words in top 100
(3): Increasing probabilities of words with high mutual information with the target concept.
(3E): Extension of (3) by simple averaging the most common words in terms of information gain of 5000 most frequent words in each class.

Table 2
Performance Measure: F1 Score
"20news-bydate" dataset
Comparison of Supervised Learning Approaches

Approach	**F1 Score**
Naive Bayes (Original)	74.4956%
Naive Bayes (Feature Selection)	83.0367%
Naïve Bayes (TF)	70.5258%
Naive Bayes (TFIDF)	77.3898%
Logistic Regression	82.7934%
Linear SVC Classifier	85.3160%
Random Forest Classifier	56.1737%

Original: Implementation of Naive Bayes
Feature Selection: Combination of (1), (2), (3E)
TF: Term Frequency
TFIDF: Term Frequency with Inverse Document Frequency

Table 3
Performance Measure: F1 Score
Cross Validation dataset
Comparison of Supervised Learning Approaches

Approach	**Averaged F1 Score**
Naive Bayes (Feature Selection)	89.3644%
Naive Bayes (TFIDF)	84.4238%
Logistic Regression	89.0295%
Linear SVC Classifier	92.4258%
Random Forest Classifier	64.4955%

Raw Data used for Table 3

Approach	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Naive Bayes (Feature Selection)	88.5292%	89.1529%	89.2882%	89.5015%	89.1195%	89.1769%	89.7012%	89.2996%	89.9683%	89.9067%
Naive Bayes (TFIDF)	83.9750%	84.2804%	84.4264%	84.2539%	84.3070%	84.3468%	84.8115%	83.8821%	84.6920%	85.2629%
Logistic Regression	88.4360%	88.6484%	88.7945%	89.3919%	88.9804%	89.2459%	89.3255%	88.8874%	89.2592%	89.3255%
Linear SVC Classifier	92.1004%	92.4854%	92.4854%	92.3128%	92.2331%	93.0165%	92.5799%	92.0074%	92.6713%	92.3659%
Random Forest Classifier	63.3962%	65.1354%	64.2857%	63.4758%	63.6617%	65.8391%	65.2151%	63.6086%	65.8125%	64.5247%