Document Classification

Course Project (CS 403)

Spring 2016

by

Jay Vora

Partha Bhattacharya

Balagopal Kanattil

Gowri Shankar

Under the supervision of

Prof. Ganesh Ramakrishnan

Department of Computer Science and Engineering, IIT Bombay

April 2016

Abstract

The problem of document classification seeks to segregate news articles, or any articles for that matter, into categories that make it easier for the content provider to provide content optimally to the consumers. For example, in the case of news subscribers, certain users maybe more interested in sports and some may prefer politics. Having documents classified into certain categories will help the news providers send appropriate news articles to the end-user. This kind of a feature will help in consumer retention and possibly even consumer acquisition. This involves two steps:

- Taking a text document and classifying into categories.
- Learning the preference of the end-user by using click-through as feedback.

For the purpose of this project, we will be focussing only on the first part, which is based on *Support Vector Machine* principles.

Contents

Chapter 1

Introduction

We began by collecting data from BBC news article repository [1]. It contains of 2225 documents from BBC. We used the pre-processed dataset provided. It contains the following files:

- **bbc.classes** : It contains the labels for each of the 2225 documents. Labels are 0,1,2,3,4 corresponding to business, entertainment, politics, sport, tech.
- **bbc.terms** : It contains the list of 9635 words that are used as features.
- **bbc.mtx** : It contains the frequency of each word in each document.

In the frequency data, each line corresponds to a row of the sparse data matrix. We converted this into a dense matrix which was used for training the model. Its code is included in make-data.py.

We followed these steps for document classification:

- Run SVM on 70 % of the data.
- Test the trained model on remaining 30 % of the data.
- Create a feature extraction script which will extract features from freshly selected news articles (not part of the original data-set)
- Predict the classes for these articles.
- Speculate and comment on methods of feature selection.

Chapter 2

Algorithms Used

2.1 Naive-Bayes

We are using Naive-Bayes module from sklearn library. The distribution used for this algorithm is Gaussian. However since the accuracy was not satisfactory, we decided to use support vector machines instead.

2.2 Support Vector Machines

We are using svm module from sklearn library with the RBF kernel[1]. We varied the regularization parameter and converged on 3,000, as it performed well on the test data.

2.3 Feature Extraction

The test data obtained from BBC came with the feature frequencies. However we want our program to be robust for articles which are supplied just like that. So we must have a pre-processing step which gathers a count of the keywords present in the article. This is done by the use of the following steps.

[1] $K(x, x') = exp - \frac{||x-x'||^2}{2\sigma^2}$

2.3.1 Tokenizing

Tokenizing is the process of:

- Removal of special symbols like '?', '!', '%', etc.
- Collecting individual words and making them elements of an array.

Once we are done tokenizing, our paragraph is now converted into an array of words.

2.3.2 Stemming

Stemming is the process of converting words into their stem words by applying certain standard transforms. A comprehensive list of those is available at [2]

Some examples of how stemming works:

- Works like walking, walks, and walked are all mapped to walk.
- For words which end with 'y', the 'y' is replaced with 'i' iff the word contains other vowels.

2.3.3 Stop word removal

Because of the nature of our features, there are several words in most articles which we do not need to keep. Such words are called stop words (eg. should, would, can, could, are, is, the, etc.). We use a list of 300 such words and reduce the size of the array.

Chapter 3

Results

3.1 For BBC test data

The accuracy on using Naive-Bayes gave 94% accuracy for 70-30 train-test split. On doing a 5-fold cross validation, we got an average accuracy of 92%.

On shifting to support vector machines, it gave a better accuracy of 98.8% in 70-30 split. On 5-fold cross validation, it gave an accuracy of 98.4%.

3.2 Testing on paragraphs outside BBC data-set

On using both algorithms, Naive-Bayes performed better than SVM on external paragraphs.

Bibliography

[1] http://mlg.ucd.ie/datasets/bbc.html