

Document Classifier

ABSTRACT

This paper covers the basics of classification techniques used to classify federalist papers. Two techniques have been discussed in brief – Rocchio classification and Naive Bayesian classification technique. Rocchio and Bayesian technique were implemented, and their results are discussed in this paper. These techniques were implemented on a common set of documents i.e federalist paper, which included training set documents and test set documents (ambiguous documents). During the comparison of these technique Bayesian showed remarkable accuracy compared to Rocchio.

INTRODUCTION

Classification plays an important role in Information Retrieval. Classification is the process of analyzing a input and assigning it to (one or more) category. Within information retrieval, classification is used in various subtasks of the search pipeline.

Test Study

To study the classification techniques, we chose the federalist papers (85 papers). This set of federalist papers contains the three classes of authors Hamilton (51 papers), Madison (15 papers) and Jay (5 papers) class. It also includes classes ‘Hamilton and Madison (contains the paper’s written by both)’ (3 papers) and ‘Hamilton or Madison (conflicting authorship)’ (11 papers). Authorship of documents in the ‘Hamilton or Madison’ class is in question as both Hamilton and Madison claims the authorship of papers. We must find who wrote these papers. To resolve the conflict, we studied and implemented classification in detail, we discuss two techniques used to classify in information retrieval system.

1. Rocchio classification
2. Bayesian classification

Rocchio Classification

Rocchio classification is a Vector based classification technique. It is based on calculating distance of the centroid of the training and test dataset. Each class has a centroid vector. The centroid calculation helps in finding a point of reference to calculate the distance with the test documents.

In Rocchio classification, first we use a training dataset to calculate the centroid vectors for each document present in the training set.

$$\bar{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

Where D_c = total number of training document of class c .

$\vec{v}(d)$ = document vector of document d which belong to D_c

After calculating centroids for all training dataset, we define regions per the classes in the training dataset. Now, we calculate centroids for individual documents in the test dataset which helps to identify the centroid of test document with respect to training dataset regions.

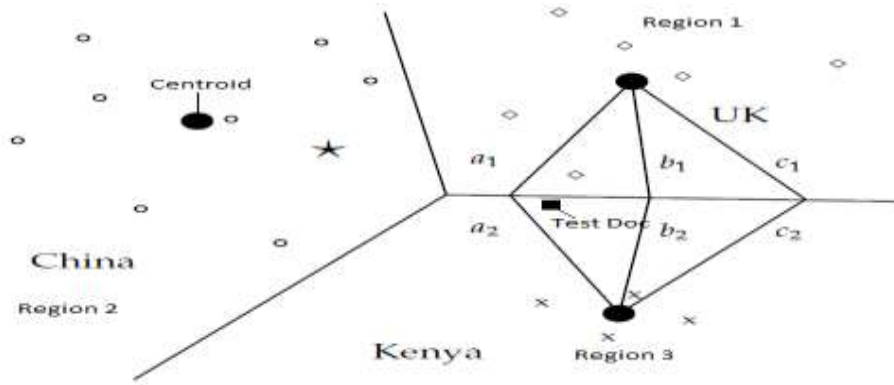


Figure 1: Rocchio Classification

Approaches to classify

Once the regions are formed, centroids of the training dataset and the documents of the test dataset are calculated, next step is to find the similarity between training and test data. To achieve this, we have two approaches in Rocchio

1. Cosine Similarity

The way of quantifying the similarity between two documents d_1 and d_2 is to compute the cosine similarity of their vector representations $\vec{v}(d_1)$ and $\vec{v}(d_2)$.

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|}$$

The minimum cosine value between training and testing dataset defines the classification of document.

2. Euclidean Distance

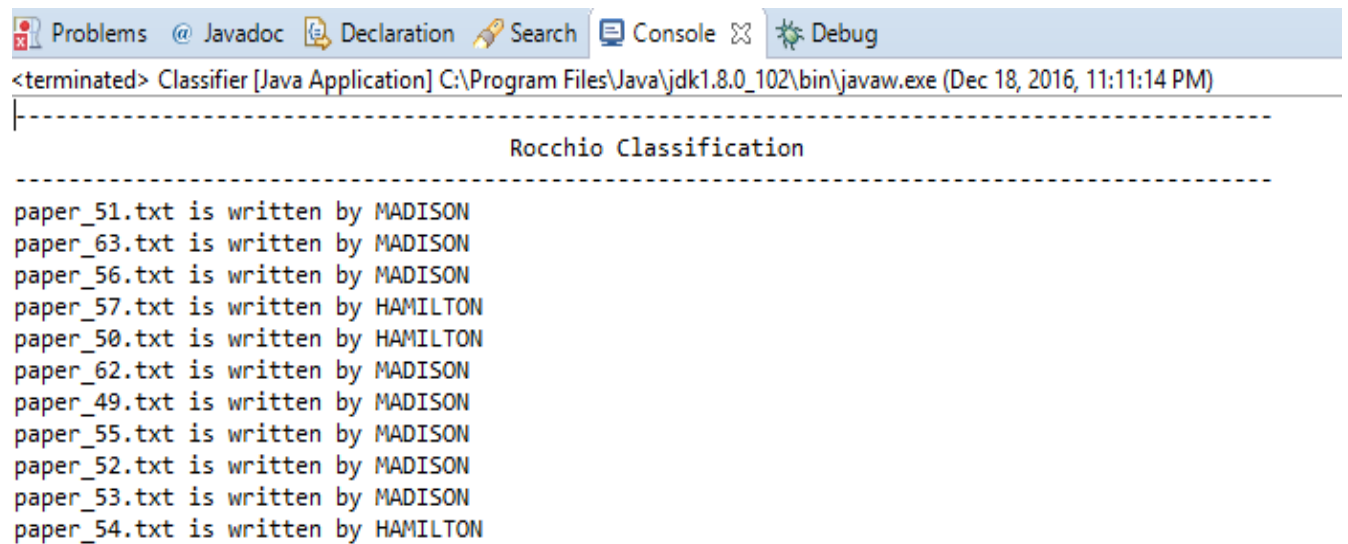
The Euclidean distance is the "ordinary" (i.e. straight-line) distance between two points in Euclidean vector space.

$$\sqrt{\sum_{i=1}^M \vec{V}_i^2(d)}.$$

The minimum Euclidean distance value between training and testing dataset defines the classification of document.

Results

Rocchio classification with Euclidean distance, classified 8 documents as Madison and 3 as Hamilton. The accuracy obtained using the algorithm is 72.7% (owing to the fact that all the documents belong to Madison class).



```

Problems @ Javadoc Declaration Search Console Debug
<terminated> Classifier [Java Application] C:\Program Files\Java\jdk1.8.0_102\bin\javaw.exe (Dec 18, 2016, 11:11:14 PM)
-----
Rocchio Classification
-----
paper_51.txt is written by MADISON
paper_63.txt is written by MADISON
paper_56.txt is written by MADISON
paper_57.txt is written by HAMILTON
paper_50.txt is written by HAMILTON
paper_62.txt is written by MADISON
paper_49.txt is written by MADISON
paper_55.txt is written by MADISON
paper_52.txt is written by MADISON
paper_53.txt is written by MADISON
paper_54.txt is written by HAMILTON

```

Figure 2: Rocchio Results

Analysis

The Rocchio classification does not achieve perfect accuracy because of its linear characteristic of dividing the space of data points linearly. As the data points for the cross category occur in disjoint clusters, it causes the classifier built by the Rocchio method to miss most of them, as the centroid of these data points may fall outside these clusters. As an example to categorize data into ‘+’ and ‘o’, as shown below, Rocchio will miss out the data points outside the cluster.

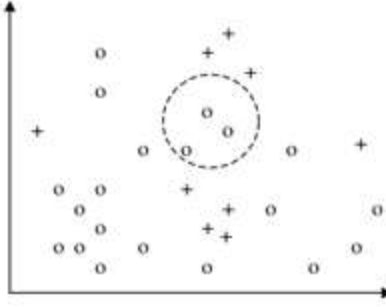


Figure 3: The representation of classification into two categories

Naive Bayesian classification

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Implementation

We calculate the probability of classes using term frequencies. For which the formula is given below

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

Where denominator is the sum of total number Documents and term frequencies of that class

The above formula on calculation result into floating point underflow leaving us with very small number. Hence to overcome this we use logarithmic function to calculate Cmap, formula given below.

$$c_{\text{map}} = \arg \max_{c \in \mathcal{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)].$$

Feature selection

To reduce the complexity and calculation in naive Bayesian we use the selective term which involve in considering limited number of important terms using below formula.

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} \\ + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}$$

Where N = total number of documents

N_{11} = number of documents written by Madison class of considered term

N_{10} = number of documents not written by Madison class of considered term

N_{01} = number of documents written by Madison class of considered term

N_{00} = number of documents not written by Madison class of considered term

$N_{1.} = N_{11} + N_{10}$

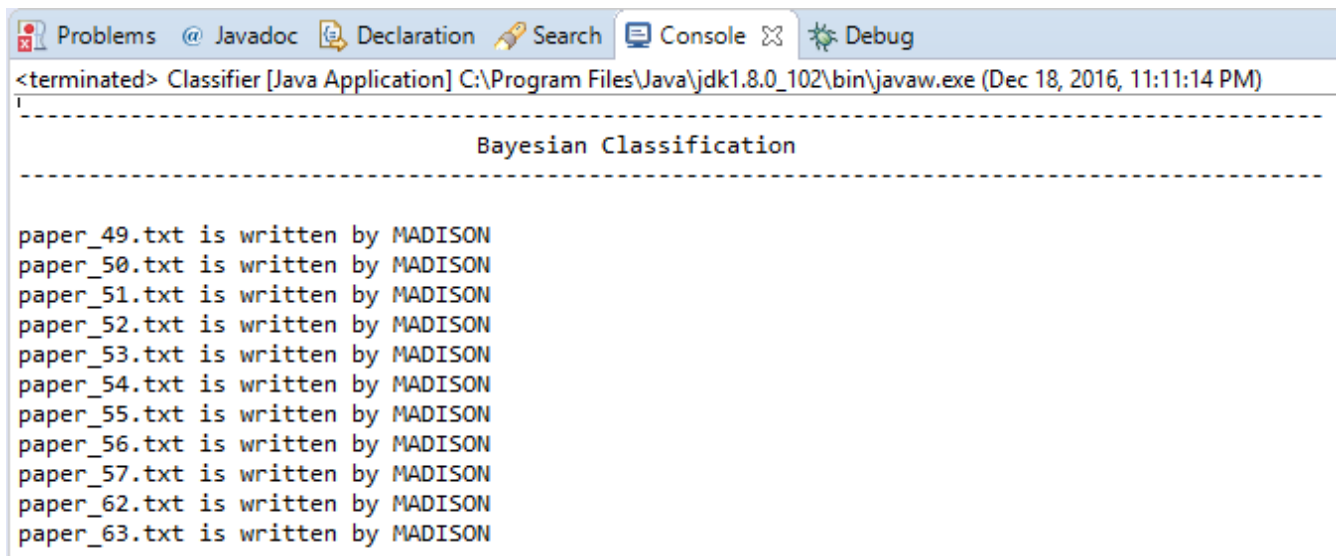
$N_{.1} = N_{11} + N_{01}$

$N_{0.} = N_{01} + N_{00}$

$N_{.0} = N_{10} + N_{00}$

Results

Using Naive Bayesian classifier, we find that all documents have been written by Madison, and hence it achieves 100% accuracy. (owing to the fact that all documents belonged to Madison).



```
<terminated> Classifier [Java Application] C:\Program Files\Java\jdk1.8.0_102\bin\javaw.exe (Dec 18, 2016, 11:11:14 PM)

-----
Bayesian Classification
-----

paper_49.txt is written by MADISON
paper_50.txt is written by MADISON
paper_51.txt is written by MADISON
paper_52.txt is written by MADISON
paper_53.txt is written by MADISON
paper_54.txt is written by MADISON
paper_55.txt is written by MADISON
paper_56.txt is written by MADISON
paper_57.txt is written by MADISON
paper_62.txt is written by MADISON
paper_63.txt is written by MADISON
```

Figure 4: Bayesian Classifier results

Analysis

The Naive Bayesian works on the assumption that the conditional probabilities are independent of each other and the terms are treated as a Bag of words. Although these assumptions are oversimplistic, Naive Bayesian classifier performs well. It maybe because we still have too many parameters for the multinomial model. In this model, we do not consider positions of a term and look at each term separately. So, the conditional independence assumption commits us to this way of processing evidence.

Conclusion

To solve the conflicting authorship of papers we studied classification techniques. First, we implemented Rocchio classifier which resulted into 72.9% accuracy in finding the authorship. Then we implemented Bayesian classification which resulted the 100% accurate results at important words ($k = 50$) and 72.9% at $k = 15$. Using the above-mentioned techniques, we find that the Naive Bayesian classifier, despite its Naive assumptions gives us better accuracy than Rocchio for text classification.

References

- [1.] Guo, Gongde, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. "Using kNN model for automatic text categorization." *Soft Computing* 10, no. 5 (2005): 423-30. doi:10.1007/s00500-005-0503-y.
- [2.] "Rocchio classification." Rocchio classification. Accessed December 18, 2016. <http://nlp.stanford.edu/IR-book/html/htmledition/rochio-classification-1.html>.