COMP90049 Knowledge Technologies Project2 Report

Wei Han (weih) 523979

Abstract

In this project, we develop a text classification software which can determine the genre of input text. The classification model used in this program is based on the Weka machine learning package. For the plain text classification, one of the most important is to extract the text features. This program develops a features extracting technique which combing the text stem frequency normalization, frequency lifting, etc. With this method, the different genres features are almost isolation each other. Combing the NaiveBayes classification model in Weka package, the 87% classification correctness for unknown text is got. With the method's further improvement and better classification model's setup, the classification correctness can be further improved.

1. Introduction

This project develops a program which can classify a text into one of a given categories set. The classifying is based on the predication model which learned from the pre-feed training text. The training texts have the similar form and contents with predication text but already been classified. The learning method should extract different categories text's features from the training materials. Based on these features, the prediction model should be set up to classify future unknown input text. In this project, the prediction model will be implemented with Weka machine learning package, so the text features extracting is more important in this project's implementation.

Although text features information can be searching with many approaches, the most basic approach is based on word info within text. The features extracting approaches used in this project are also based on the analysis for the words statics info within text. The techniques used include:

- text tokenization
- stop words removing
- words stemming
- words frequency counting
- frequency normalization
- frequency lifting
- genre based features words extracting
- ...

With the implementation of these techniques, the project's final approach can almost achieve features isolation between different categories, i.e., one category's feature word almost no existing in other category's features list[1].

This project's classification model is based on Weka machine learning package. After the system features list has been built, all the training and testing files can be featured (i.e., set-up feature vector for each file) and then corresponding ARFF file can be created. Feed these ARFF files into Weka, we can evaluate different prediction model's performance, such as accuracy, precision, etc. Then we can select the best model for the project. After these comparisons, this project finally selects the Naive Bayes classification. After training with the texts in the directory /train, the program can achieve the 87% classification correctness for the texts provided in /dev directory.

2. Text features extracting

As mentioned above, the technical approach in this project includes two steps: text features extracting and classification model building. In this section, we will give some detail discussion for text features extracting.

The basic idea for text features extracting in this project is that the words frequency has the different distributions in the different text genre. So the features extracting is how to find those words which just common in one genre, but rare in other genres. All the features extracting techniques used this project is try to find and notification this differences.

2.1 Text Tokenization and Stemming

To find above mentioned feature words, the first step should be text tokenization, i.e., splitting the text into the set of single words. Also in this step, every word's frequency will be count during the text splitting. For some words are very common and they don't show the notable difference between different genres, their high occur frequency probably will disturb the features extracting accuracy. So the second step is to

[1] Our final results show all 1200 features are unique when select 150 features for each genre. Only 3 duplications when select 200 features for each genre.

remove these words, this is called stop words removing.

A word can appear in different forms in the different occasions, but they still retain the similar means, the stemming is to reduce the word into its stem from. This project uses the Porter Stemmer to stem to strip the suffix of the words. During the stemming process, the same family's word frequencies are also merged into the word stem.

2.2 Stem Frequency Normalization

When we merging different text's word frequency, we can see that a word in large text may be have large occurrence count, even it not common in that text genre, i.e., not have high occurrence count in other text files in same genre. And the feature candidate word in short text file has the possibility to omit for the simple adding. To overcome this problem, before merging different file's words frequency, the program will normalize the word occurrence count into the occurrence number in the unit text length, i.e. transforming the simple word occurrence count into the word occurrence frequency. The adding these frequencies could get the stem total frequency and word genre occurrence frequency.

2.3 Stem Frequency Lifting

When we creating the features set, we would like those feature words just appear in one category, and don't appear in other categories. So the stems which have the higher occurrence frequency in special category but low frequency in other categories should be a better candidate than those stems which have the evener frequency distribution even they have a higher total frequency. To lift those better candidates' frequency, this project use the following equation to adjust all the stems' frequency:

$$Freq_{new} = Freq_{old} * \left(\frac{7 * Freq_{old}}{Freq_{total} - Freq_{old}}\right)^{r}$$

Where $Freq_{new}$ and $Freq_{old}$ are the stem frequency in a special genre, $Freq_{total}$ is the stem frequency in the total training material. So $\frac{7*Freq_{old}}{Freq_{total}-Freq_{old}}$ can be called **lift coefficient** for a given stem. It reflects the stem frequency distribution's uneven degree in indifferent garners. The coefficient *r* can be used for further adjustment.

2.4 Features Set Creating

So far, the mainly techniques used to count stem frequency have been discussed. The last step is the features set creating. Because, so far we have created all the categories' candidate stems and their corresponding frequency, we just pick the most frequent stems from each category and put them into one set, then can get the final features set.

The final results show our above features extracting approach almost come to our initial target—genre features isolation. When we select r>1.4 in lift frequency calculating and select 200 most frequent stems for each genre to build a final features set, we find the final set size is 1600. This means no duplication in this case. For our final r setting r=1.3, still 200 most frequent stems for each genre, the final features set size is 1597, this means just 3 duplication occur.

3. Classification model selection:

For the classification model used in this project is based on Weka machine learning package, the technical approach in the classification is focused on the comparison and selection of different models. After some initial testing and comparison, we find the following three Weka models deserve further comparison for the stuff in this project: J48 decision tree, NaiveBayes classification, ComplementNaiveBayes classification.

In our project developing versions testing, where the genre features number is relatively small (~100 or smaller), and training files number also smaller than 300 for speed developing and testing, all these three methods have some best performance in some cases. But with features number's and training instances' increase, although all these three model's classification correctness are increasing, the Naive Bayes method show the better performance. For the project's final version and parameters setting, using full training files and testing all dev files, the NBC achieved the 87% classification correctness, while J48 is 72%, CNB is 76%. The more detail class accuracy for the final version program is:

J48:

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
0.72	0.05	0.703	0.72	0.7	0.881

NBC:

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
0.87	0.017	0.86	0.87	0.857	0.975

CNB:

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
0.76	0.102	0.769	0.76	0.735	0.829

From above detailed comparison, we can see for our final features selection, NBC has the absolute best performance. It has the best TP-Rate,Precison and Recall. And the lowest FP rate. So the NaiveBayes classification is selected as the

program's final classification implementation.

CNB is a little bit better than J48. Its TP-Rate, Precision and Recall all slightly higher than J48, but its FP-rate also higher than J48.

The reason the statistics based NaiveByes Classification performances best in the more attributtes and more instances condition is that with the growth of statistical sample, the statistical accuracy will also increase. On the other hand, the decision tree based methods, such as J48, is hard to take this advantage.

4. **Testing:**

Implementing above features extracting and classification model selection into the program, training with /train files, then classify the files in /dev directory, we can get following classify results:

'Dev.class' file	'results.class' file	Correct? (Y/N)
25110-0.txt,E	25110-0.txt,E	Y
pg10743.txt,A	pg10743.txt,D	N
pg11583.txt,G	pg11583.txt,G	Y
pg12192.txt,H	pg12192.txt,H	Y
pg1298.txt,H	pg1298.txt,H	Y
pg13054.txt,D	pg13054.txt,D	Y
pg13316.txt,E	pg13316.txt,E	Y
pg13375.txt,D	pg13375.txt,D	Y
pg13604.txt,B	pg13604.txt,B	Y
pg14301.txt,G	pg14301.txt,D	N
pg14735.txt,F	pg14735.txt,E	N
pg15580.txt,H	pg15580.txt,H	Y
pg1563.txt,C	pg1563.txt,C	Y
pg15651.txt,D	pg15651.txt,D	Y
pg15778.txt,B	pg15778.txt,B	Y
pg16196.txt,D	pg16196.txt,D	Y
pg1642.txt,E	pg1642.txt,E	Y
pg16833.txt,E	pg16833.txt,E	Y
pg17028.txt,G	pg17028.txt,G	Y
pg17959.txt,C	pg17959.txt,C	Y
pg18257.txt,G	pg18257.txt,G	Y
pg1874.txt,B	pg1874.txt,B	Y
pg18786.txt,G	pg18786.txt,G	Y
pg1896.txt,D	pg1896.txt,D	Y
pg19090.txt,G	pg19090.txt,G	Y
pg19246.txt,H	pg19246.txt,H	Y
pg19726.txt,G	pg19726.txt,G	Y
pg20043.txt,B	pg20043.txt,B	Y
pg20651.txt,B	pg20651.txt,B	Y
pg20887.txt,E	pg20887.txt,E	Y
pg20920.txt,G	pg20920.txt,G	Y
pg21240.txt,B	pg21240.txt,H	N
pg21298.txt,D	pg21298.txt,B	N
pg2162.txt,F	pg2162.txt,E	N
pg21681.txt,B	pg21681.txt,B	Y
pg21720.txt,B	pg21720.txt,B	Y
pg21988.txt,G	pg21988.txt,G	Y
pg22512.txt,G	pg22512.txt,G	Y
pg22544.txt,G	pg22544.txt,G	Y
pg22596.txt,G	pg22596.txt,G	Y
pg22876.txt,G	pg22876.txt,G	Y
pg23028.txt,G	pg23028.txt,G	Y
pg23161.txt,G	pg23161.txt,G	Y
pg23292.txt,B	pg23292.txt,B	Y
pg23335.txt,G	pg23335.txt,G	Y
pg23571.txt,G	pg23571.txt,G	Y
pg23799.txt,G	pg23799.txt,G	Y
pg24064.txt,G	pg24064.txt,G	Y
pg24180.txt,G	pg24180.txt,G	Y
pg24247.txt,G	pg24247.txt,G	Y
pg24436.txt,G	pg24436.txt,G	Y
pg24707.txt,G	pg24707.txt,G	Y
pg24973.txt,G	pg24973.txt,G	Y
pg2509.txt,G	pg2509.txt,G	Y
pg25776.txt,G	pg25776.txt,G	Y
pg26206.txt,G	pg26206.txt,G	Y
pg26882.txt,G	pg26882.txt,G	Y
pg27110.txt,G	pg27110.txt,G	Y
pg274.txt,E	pg274.txt,E	Y
pg2759.txt,A	pg2759.txt,D	N
pg28048.txt,G	pg28048.txt,G	Y
pg28550.txt,G	pg28550.txt,G	Y
pg28698.txt,G	pg28698.txt,G	Y
pg29059.txt,G	pg29059.txt,G	Y
pg29195.txt,G	pg29195.txt,G	Y
pg29384.txt,G	pg29384.txt,G	Y
pg29488.txt,G	pg29488.txt,G	Y
pg29578.txt,G	pg29578.txt,G	Y
pg29698.txt,G	pg29698.txt,G	Y
pg29822.txt,G	pg29822.txt,G	Y
pg29963.txt,G	pg29963.txt,G	Y
pg30015.txt,G	pg30015.txt,G	Y
pg30214.txt,G	pg30214.txt,G	Y
pg30311.txt,G	pg30311.txt,G	Y
pg30866.txt,E	pg30866.txt,E	Y
pg3139.txt,D	pg3139.txt,D	Y
pg369.txt,G	pg369.txt,D	N
pg4014.txt,C	pg4014.txt,C	Y
pg4057.txt,D	pg4057.txt,E	N
pg4320.txt,E	pg4320.txt,E	Y
pg4724.txt,E	pg4724.txt,E	Y
pg4922.txt,H	pg4922.txt,H	Y
pg4989.txt,B	pg4989.txt,B	Y
pg5075.txt,D	pg5075.txt,D	Y
pg5520.txt,D	pg5520.txt,D	Y
pg599.txt,D	pg599.txt,B	N
pg60.txt,A	pg60.txt,A	Y
pg624.txt,G	pg624.txt,E	N
pg6366.txt,E	pg6366.txt,E	Y
pg6461.txt,B	pg6461.txt,B	Y
pg6762.txt,F	pg6762.txt,E	N
pg6940.txt,D	pg6940.txt,D	Y
pg7205.txt,E	pg7205.txt,C	N
pg7720.txt,D	pg7720.txt,D	Y
pg7896.txt,C	pg7896.txt,C	Y
pg836.txt,B	pg836.txt,B	Y
pg9267.txt,H	pg9267.txt,H	Y
pg9785.txt,D	pg9785.txt,D	Y
pg9862.txt,G	pg9862.txt,G	Y
pg989.txt,E	pg989.txt,E	Y

Comparing above table left side's input 'dev.class' file and the middle side's output 'results.class" , we can see for 100 unknown input files, there are 13 files classified wrong. The classify correctness is 87%, this is the same as the with Weka GUI's analysis.

5. **Conclusions**

In this project, we developed a text classification software based on Weka machine learning package.

For the text file classification, one of the most important tasks is to extract the suitable features. In this project, we developed a word based text file features extracting technique. This feature extracting process currently including following steps:

- text tokenization
- stop words removing
- words stemming
- words frequency counting
- frequency normalization
- frequency lifting
- genre based features words extracting
- ...

With this technique, the different genre feature isolation is almost achieved. Combing the NaieveBayes classification in Weka package, 87% classification correctness for unknown text file is got.

With the further improvement of methods and the better classification method's setup, the higher classification correctness can be achieved.

References

[1] George Forman, Feature Selection for Text Classification. Computational Methods of Feature Selection Chapman and Hall/CRC Press, 2007

[2] Sam Scott, Stan Matwin, Feature Engineering For Text Classification

[3] Ian H. Witten, Eibe Frank, Data Mining Practical Machine Learning Tools and Techniques, Second Edition, 2005 by Elsevier