CSE 5243-Assignment I

Group Member: Fang Zhou

1. Problem Description

In this assignment, the objective is to preprocess the data, get refined data composed of feature vector and class label(s) for each article in the Reuter dataset. This is the first step of our term-long project. This is very important, because if we cannot preprocess or prepare the important information well from the large scale of data included in each document, it is very hard to reach our final objective.

2. Structure of The Original Reuter File

The Reuter data set includes several Reuter files. A Reuter file includes one or more Reuter documents. Actually, the Reuter file is a standard xml document with a lot of meaningless characteristics and words. Excluding the unmeaning characteristics, we get a standard xml document. As we mentioned at the beginning, one Reuter file includes one or more Reuter document with Reuter tag. In each Reuter document, there are several tags: date, topic, place, people, orgs, exchanges, companies, and text. Text part contains title, dateline, and body. The body part is composed of the main words in this document. So this part is the most important factor to compare the similarity.

3. Approach

In this section, I will introduce my approaches to get the feature vectors from the Reuter dataset. The first step is to preprocess the data to make it clean. The second step is to parse the data. The third step is to extract the feature vectors in two levels, fine grained and coarse grained. Totally, I have three feature vectors. In the fine grained level, I extract two features based on word frequency and TF-IDF. In the coarse grained level, I get the features based on the general information.

3.1 Preprocess

There are a lot of meaningless data in the Reuter files. Before we can process the data further more, we should first clean the data. A Reuter file's main structure is a xml file. So the first step is to remove the unaccepted characteristics (&, <, and >) for the xml parser. After this step, we can use any xml parsers to parse the file successfully. For the coarse grained features, we can directly extract the vectors based on the main tag. However, for the fine grained features, we still need to process the text further. After removing the invalid characters, we should continuously remove the "stop word" in the text. This kind of words cannot provide us more semantic information and may lead a misunderstanding of the content. After processing the "stop word", we should use stemmer to process the text to reduce the scope of the text. Stemmer is a process for removing the commoner morphological and flexional endings from words in English.

3.2 Parse

After preprocessing the data, we can use one xml parser to help parse the data. The id and the body part are the most important to extract.

3.3 Feature Vectors

We want to use feature vectors to represent a document. I use newid and oldid to identify one document, which are extracted from the Reuter file. Except these two ids, rest each attribute in one vector show one feature of this particular document. I will first introduce the fine grained feature vectors used in my project. Then I will present the coarse grained feature vectors.

3.3.1 Fine Grained Feature Vectors

Fine grained feature vectors are generated based on the document content without any other information, such as title, date, and so on. We use words inside body part to analyze.

Feature Vectors based on The Word Frequency

The most intuitive method to get the feature vectors based on the word frequency. The step is to collect the word frequency based on the combination set of words from all the Reuter documents. Then we will use the top n[1] words as the attributes for every document in the file. After getting the top n words, we will calculate the frequencies of the n words in a specific document and use these values as the vector's attribute values.

Feature Vectors Approach based on the TF-IDF

The other fine grained method is to use TF-IDF to process the text. The full name of TF-IDF is term frequency–inverse document frequency. TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document. I choose one kind of TF and IDF functions shown below.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
IDF(t) = $\log_e$(Total number of documents / Number of documents with term t in it).

This feature vector is designed to have the same attributes with the previous one. So I compute the values for these attributes based on these two formula.

3.3.2 Coarse Grained Feature Vectors

Fine grained feature vectors should be very good vectors to differentiate two documents. However, it costs a lot of time. If we release the conditions, which means we want to compare the documents in a higher and more general level, we can directly use the attributes in a Reuter file to generate coarse grained feature vectors and save a lot of resource.

Feature Vectors based on the general information

In order to get these kind of feature vectors, I extract the general information of each document from the dataset. The attributes include topic, id, date, title, and so on. If there is empty for an attribute, the program set "NULL" to the attribute. The more details can be seen in the code and readme file.

Implementation

I use Python to implement this project in standalone mode. I choose several Python libraries to help me to process the data. I use NLTK to remove the "stop word", get tokens of words, and

[1] "n" is a value that user defines. In this assignment, I set the value to 100.

run stemmer processing. I also use BeautifulSoup4 to parse the Reuter file and NLTK to calculate the word frequency. I create several classes so that the structure of this project is systematic and easy to understand and expand. The expected input of the project is a Reuter dataset directory and the output are three files for three kinds of feature vectors, respectively.

Evaluation

Here, I use some figures to show some results (n=100) processed by my program. Figure I is the result of feature vectors based on the word frequency. Figure II is the result of feature vectors based on TF-IDF. Figure III is the result of feature vectors based on general information.

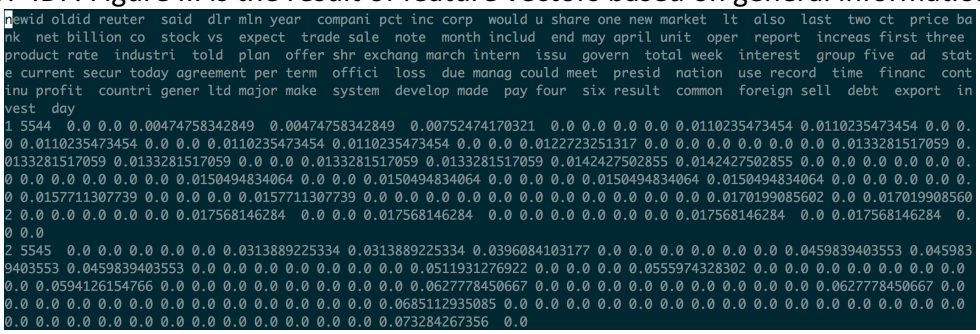

Figure I: Results based on TF-IDF

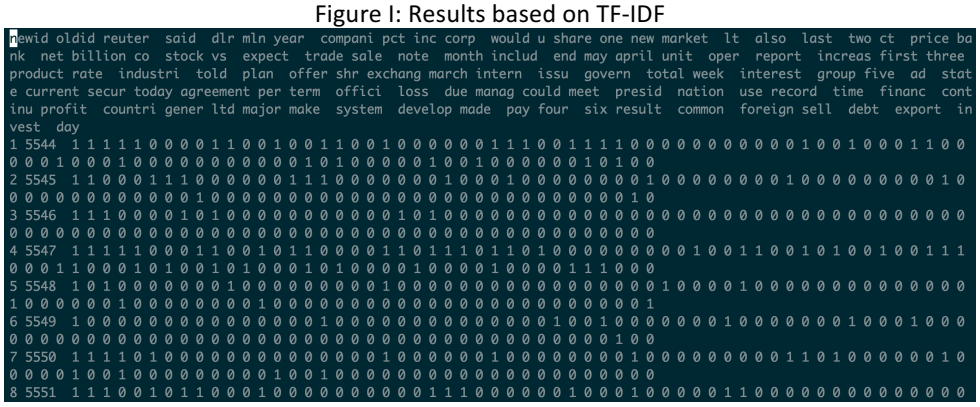

Figure II: Results based on word frequency

Figure III: Results based on general information

From these three figures, we can see, if very easy to understand the meaning of these three results. In each result, the first line shows the attribute names. The rest lines show the feature vector's values of each document in the data set.

Conclusion

This is the first step of our term-long project. I provide three different kinds of feature vectors for future usage. In this project, I learn how to preprocess the data, parse the data, and analyze the data. Meanwhile, I also learn how to use Python libraries so that I feel the power of Python.