Midterm Project Report for EECE5642

Haoran Ding
Master student of ECE
+1 857-265-0095
ding.h@husky.neu.edu

Zhengang Li
PhD student of ECE
+1 617-331-9314
li.zhen@husky.neu.edu

ABSTRACT
In this report, we present our analysis for 20 Newsgroups dataset, including the visualization results, statistical information and clustering results based on different pre-processing ways. We use some text processing methods like Bag-of-words, TF-IDF, LDA and Doc2Vec to convert the raw text into numerical vector format, and apply K-means clustering algorithm and use visualization tools to present the results.

Keywords
20 Newsgroups, Bag-of-words, TF-IDF, LDA, Doc2Vec, K-means.

1. INTRODUCTION
In our project, we focus on text document, and our goal is to visualize different documents through pre-processing and text processing methods.

2. STATISTICAL INFORMATION
2.1 Dataset Information
We choose 20 Newsgroups dataset as our analysis target. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of my knowledge, it was originally collected by Ken Lang, probably for his *Newsweeder: Learning to filter netnews* paper, though he does not explicitly mention this collection. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering [1].

In order to facilitate the program to read data, we directly use the PyThon API provided by sklearn.datasets.fetch_20newsgroups.

The dataset has a total of 11,314 articles, divided into 20 categories. We count the proportion of each category of article, and use a pie chart to illustrate the result.

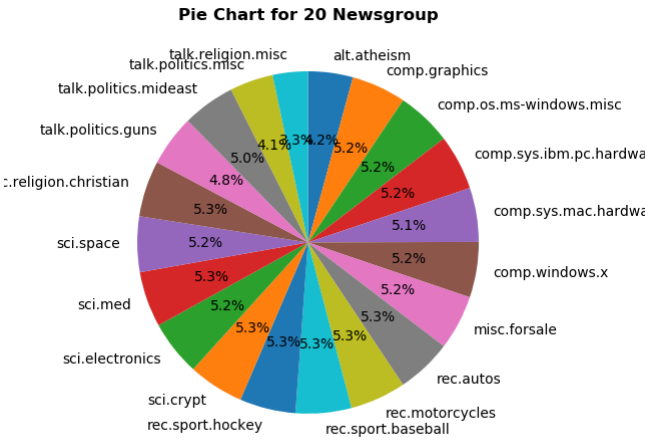

Fig 2.1.1

2.2 Vocabulary Information
Since the vocabulary may have a vital influence on the following work, it is necessary to visualize the vocabulary information.

Firstly, we apply a pre-processing on the raw dataset, including removing all the punctuations, removing numbers and non-alpha words, and converting the text into lowercase. By the way, for the punctuation apostrophe, we choose to remove it rather than replace it by a space. For example, the word “don’t” after our pre-processing, it would become “dont” because we want keep its original meaning after pre-processing.

Then we use the API CounterVectorizer provided by sklearn to perform word frequency statistics and filtering the words. By setting the parameters “max_df” and “min_df”, we can easily control the size of the vocabulary. For example, setting “max_df=0.9” means the program would drop the words with document frequency bigger than 0.9, and setting “min_df=0.1” means that the program would drop the words with document frequency smaller than 0.1. Here are some histogram results for vocabulary when we change the API parameter “max_df”:

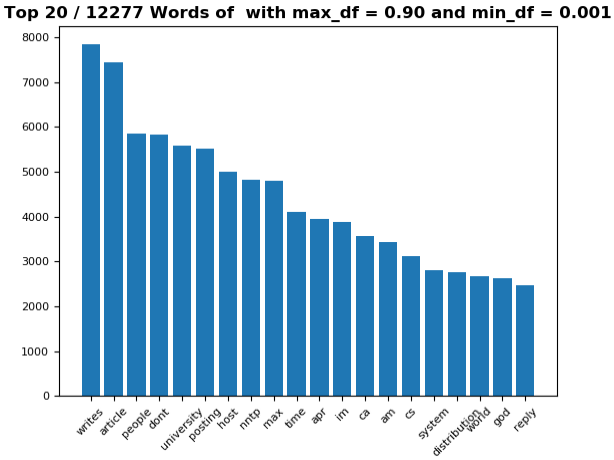

Fig 2.2.1

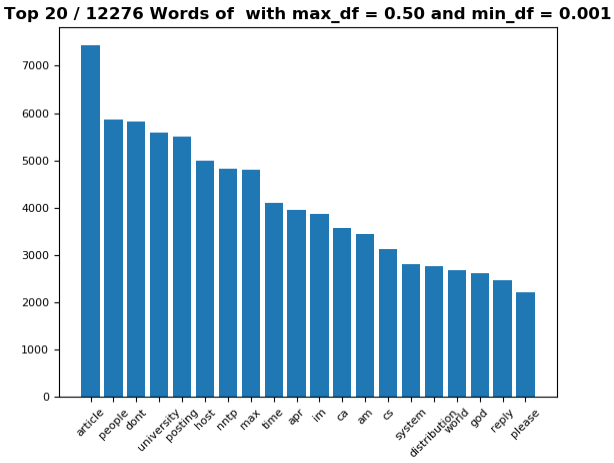

Fig 2.2.2

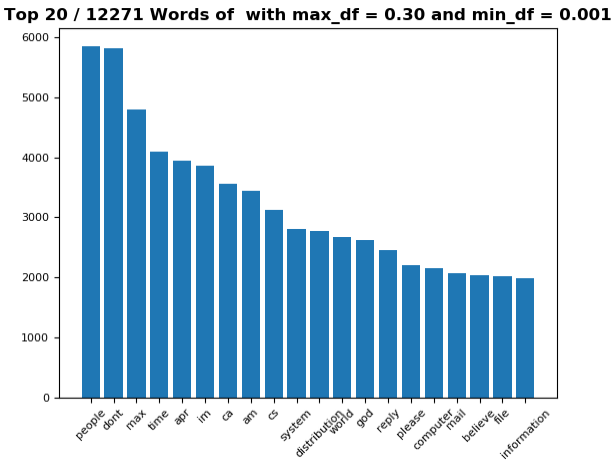

Fig 2.2.3

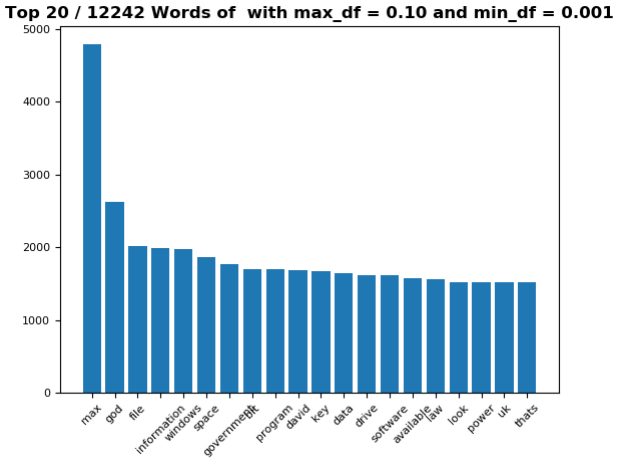

Fig 2.2.4

Combined with the definition of the API parameters and the results of the above figures, we can draw a conclusion that when we change the “max_df” from 0.9 to 0.1, the top 20 words would have a significant change while the vocabulary size will not change too much. That is because the program will only drop the high frequency words, which is really like the removing stop-words operation.

Here are the visualization results for vocabulary when we change the API parameter “min_df”:

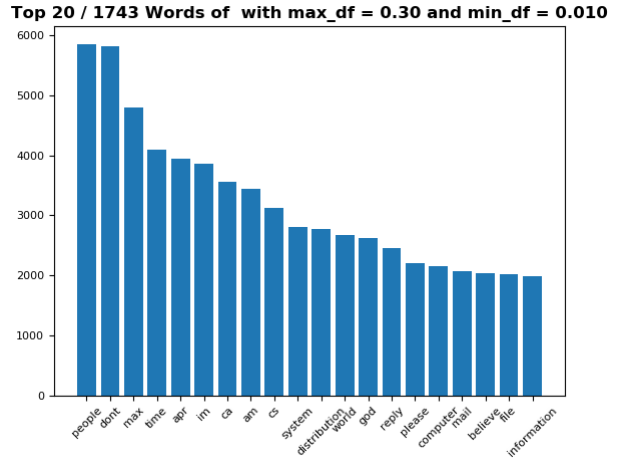

Fig 2.2.5

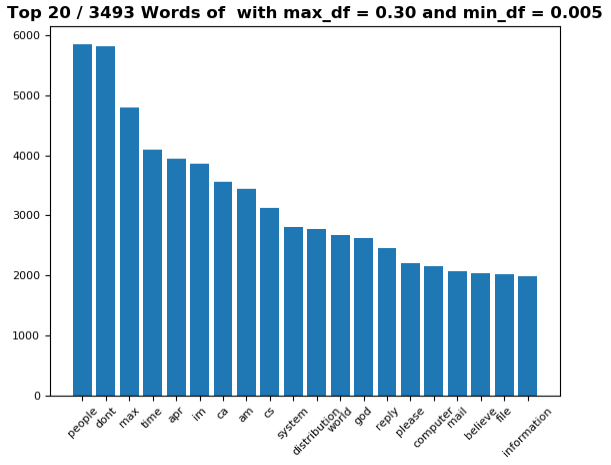

Fig 2.2.6

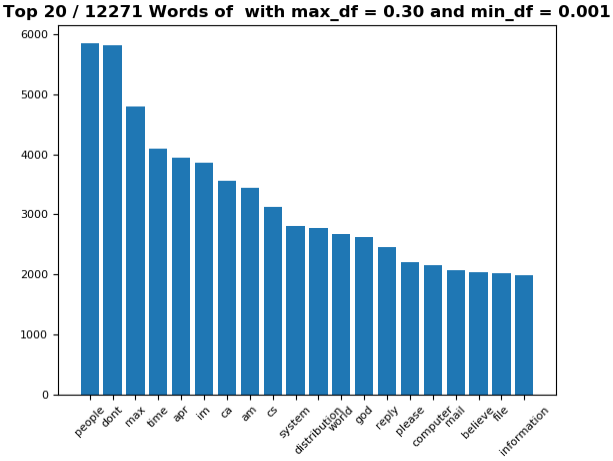

Fig 2.2.7

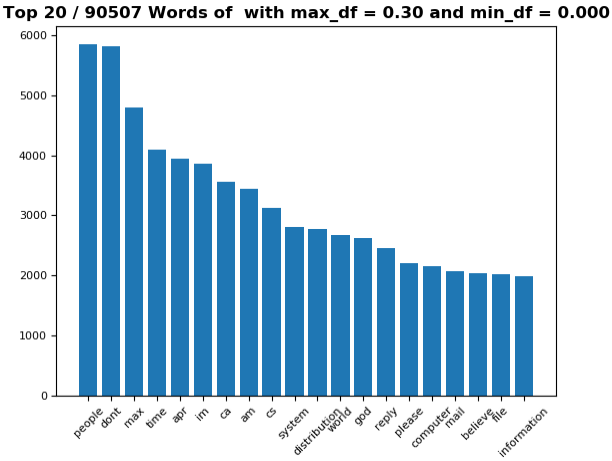

Fig 2.2.8

According to the above figures, we know that the API parameter "min_df" is the decisive role for vocabulary size.

After this experiment, we find that we can easily control the vocabulary size by setting 2 parameters.

Then we perform BoW and TF-IDF on the raw dataset. We think the purpose of BoW and TF-IDF is to transform the text format data into numerical vectors, so that we can perform other algorithms like K-means and logistic regression based on these results.

3. Latent Dirichlet Allocation Model

3.1 The Principle of LDA Algorithm

Topic modeling is a technique to extract the hidden topics from a large volume of documents. Latent Dirichlet Allocation (LDA) is a popular algorithm for topic modeling. We use both the Python sklearn's and gensim's implementation version.

In LDA, each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. LDA is a generalization of the pLSA model, which is equivalent to LDA under a uniform Dirichlet prior distribution [2].

3.2 Visualization Results

We use the sklearn API to train the LDA model. The text after pre-processing is given to the LDA model and the topic number should be set by hand. For the convenience of display the visualization result, we choose 5 categories of article for training LDA model. Considering that the number of categories is certain, we set the topic number to 5. And we perform LDA algorithm on different text representation method, like BoW and TF-IDF, and then we generate word cloud to visualize the topic distribution result.

Here are the word cloud results for Bag-of-word processing way:

Fig 3.2.1

Here are the word cloud results for TF-IDF processing way:

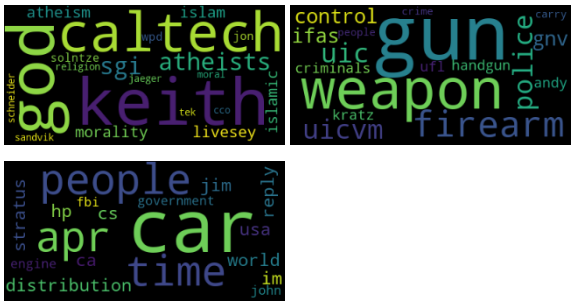

Fig 3.2.2

Then we use the LDA visualization tool pyLDAvis to visualize the result.

Since the result of pyLDAvis is a html file, we use an example to explain the result.

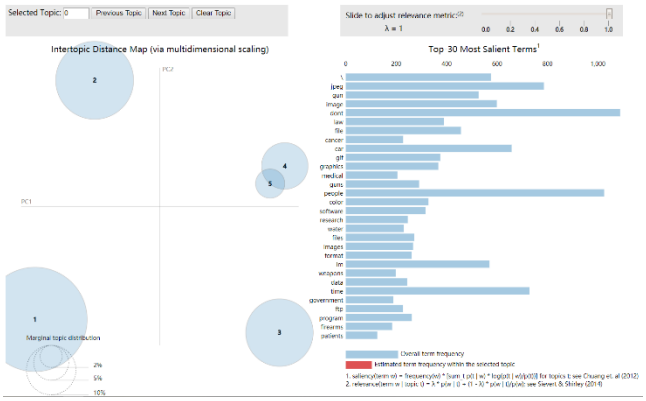

Fig 3.2.3

Fig 3.2.4

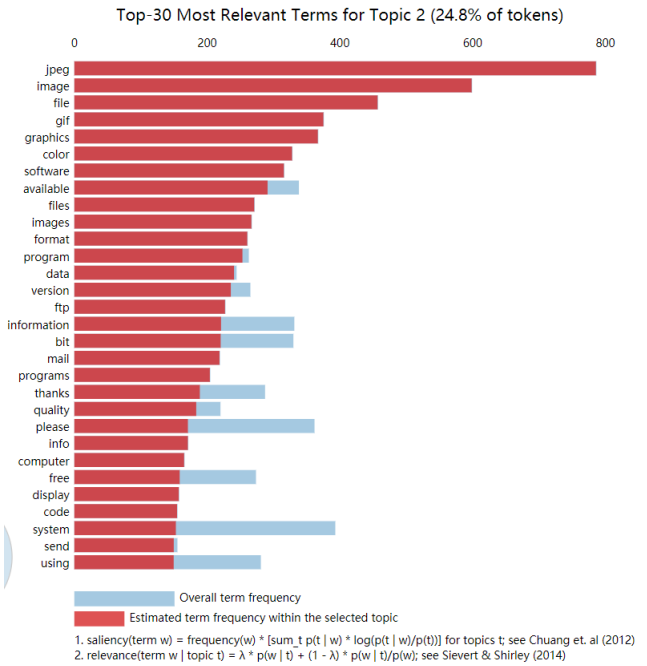

Fig 3.2.5

For example, in the Fig 3.2.3, there 5 circles on the left side. That means there are 5 topics, and each circle represents one topic, and the larger the circle is, the more documents belongs to this topic. In Fig 3.2.4, when we choose Circle 2, the Top-30 most relevant terms would be in highlight in Fig 3.2.5. According to the result, we can infer that the topic 2 would be the category "computer.graphics".

4. DOC2VEC

Doc2Vec is an unsupervised algorithm to generate vectors for sentence/paragraphs/documents [3]. The goal of Doc2Vec is to create a numeric representation of a document, regardless of its length.

We use genism tool to train the Doc2Vec model, and use t-SNE to visualize the result by generating a scatter diagram.

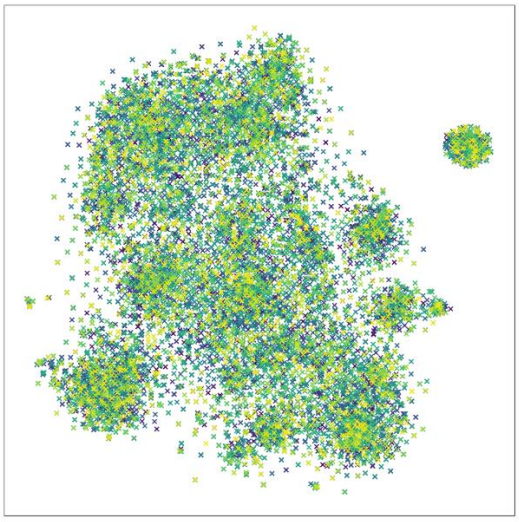

Fig 4.1 Scatter Diagram of Doc2Vec

5. K-MEANS CLUSTERING ALGORITHM

K-means is an unsupervised clustering algorithm. We perform k-means on the vector representation obtained by BoW of TF-IDF or Doc2Vec, and we use inertia as the metrics to evaluate the clustering results. Inertia is the sum of distances of samples to their closest cluster center.

We use t-SNE to visualize the result and here are the results:

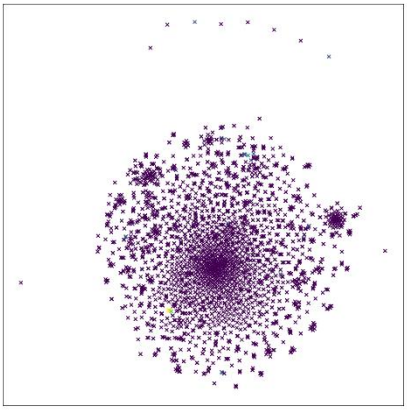

Fig 5.1 K-means Result for BoW

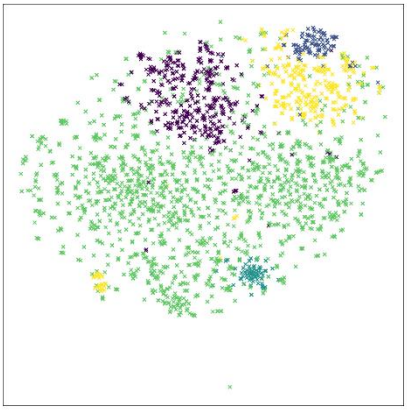

Fig 5.2 K-means Result for TF-IDF

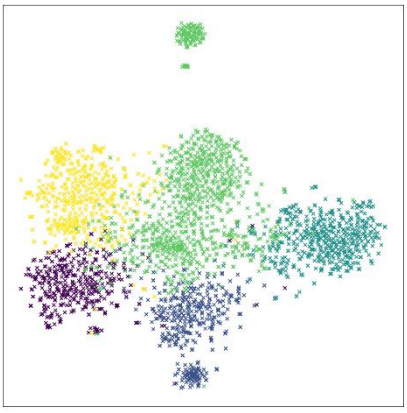

Fig 5.3 K-means Result for Doc2Vec

According to the above figures, the result of Doc2Vec is the best. We think that the Doc2Vec model can extract more features during the training because it can discard some useless features.

6. ANALYSIS

According to the above experimental results, we can draw some empirical conclusions for our project.

6.1 Impact of Different Pre-processing Ways

For an NLP problem, the vocabulary can have a vital influence on the final result so that how to build a vocabulary is really important.

According to our results and other scholars' opinion, the size of vocabulary is not the bigger the better. At the beginning, we try to keep as many words as possible to prevent loss of feature information. However, the it turns out that the result is not the case. For our dataset, a 5000 words vocabulary is enough for the feature engineering and other works.

6.2 Impact of Different Topic Numbers

In our case, the number of topics is curtained because we select the certain number of topics. However, in most cases the number is not curtained. In order to determine the appropriate number of topics, we need to calculate the perplexity of the LDA model. The formula is

$$\text{perplexity}(D_{test}) = \exp\{\frac{-\sum_{d=1}^{M}\log(p(w_d))}{\sum_{d=1}^{M}N_d}\}$$

Then we can plot a line chart like this:

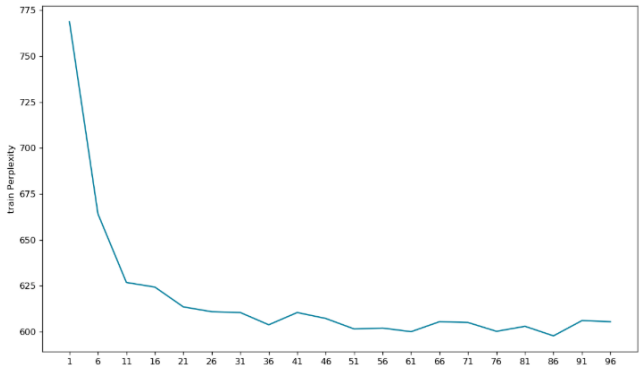

Fig 6.2.1 Perplexity Chart [4]

6.3 Different Training Methods for Doc2Vec

There are 2 methods to train a Doc2Vec model which is very similar to word2vec model.

The first method is called Distributed Memory Model of Paragraph Vectors (PV-DM), which is similar to the CBOW model in word2vec.

Another method is called Distributed Bag of Words of Paragraph Vectors (PV-DBOW), which is similar to the skip-gram model in word2vec.

6.4 The Key Factor for Doc Visualization

We think the most import thing is to determine the purpose of the visualization. For example, our target is to visualize the result of k-means so that we would draw a scatter diagram to show the clustering result.

In a word, the key factor we think is the target we want present to the viewers.

7. REFERENCES

[1] 20 Newsgroups Dataset
http://qwone.com/~jason/20Newsgroups/

[2] LDA explanation
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[3] Doc2Vec
https://www.quora.com/What-is-doc2vec

[4] Perplexity Chart
https://blog.csdn.net/u014449866/article/details/80218054