Natural Language Processing with Supervised and Unsupervised Method

Group 6:
Tianyi Liu
Haoning Xia

Introduction

Natural language processing is becoming important tools to understand human language and to perform analysis to understand people's behavior.

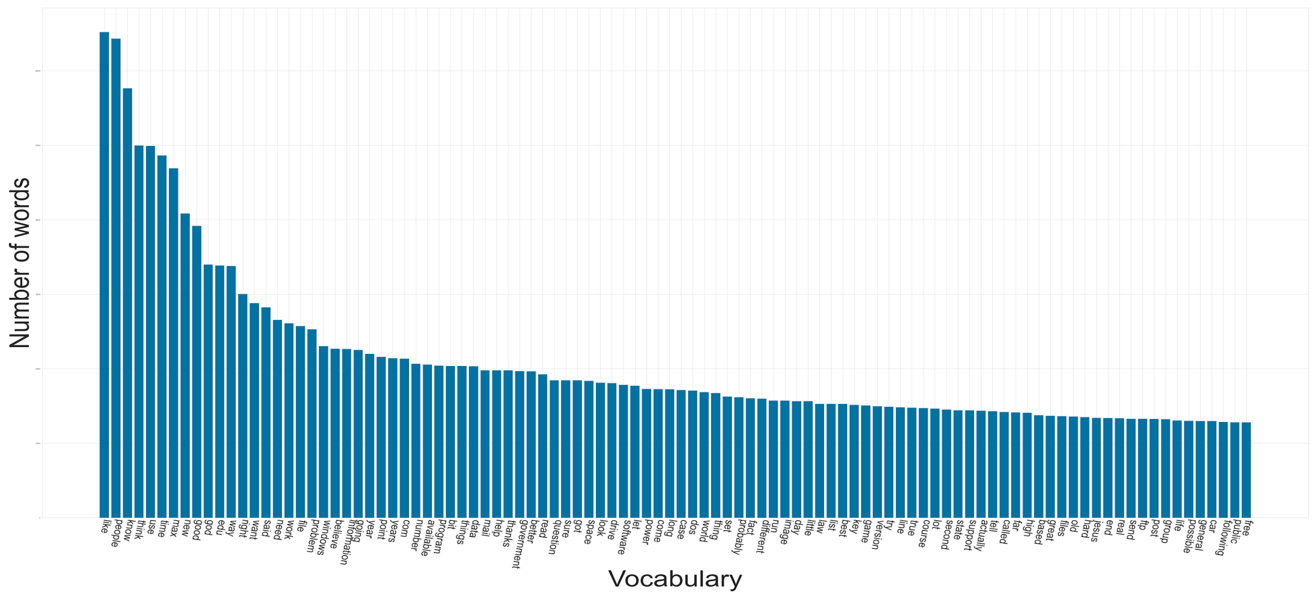
In order for machines to understand human language, we need to transform human language to machine-readable data. Therefore, the first step of natural language processing is to use different document representation methods to translate human language into data. In this project, we used a variety of different document representation methods including Bag-of-words, TF-IDF, LDA, Word2Vec, and Doc2Vec. And Kmeans, SVMs, RNNs, Naive Bayes clustering methods.

Data Preprocessing

For this project, we used 20 Newsgroups as the dataset. 20 Newsgroups is a collection of 18846 data entries across 20 different newsgroups (or topics) and each group corresponds to a different topic. In addition, all those 20 different newsgroups can be sorted into 6 larger subjects.

Before we use any document representation method, we need to preprocess the raw data. First of all, we remove all the headers, footers, and quotes from the original raw data. Then we just simply use the gensim built in preprocessing method (preprocess_string) to tokenize and filter including tags, words with length less than 3, all punctuation, digits, white spaces, and change all characters to lowercase.

After data preprocessing, we get word frequency for remaining data. Figure below shows the top 100 most frequent words from the data. Since the majority of the terms in this graph are nouns and verbs . We can conclude that we have a fairly decent preprocessing method.



Document Representation

Bag-of-Words

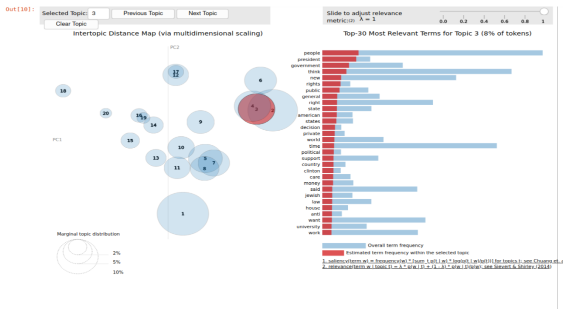
Bag-of-words model is a simple document representation method (we can simply use the Gensim method doc2bow to achieve this). In this model, we turn the text into a bag of its words. We do not care about grammar and the original word order. Bag-of-words model is a pretty common and simple method for document classification.

Term Frequency and Inverse Document Frequency(TF-IDF)

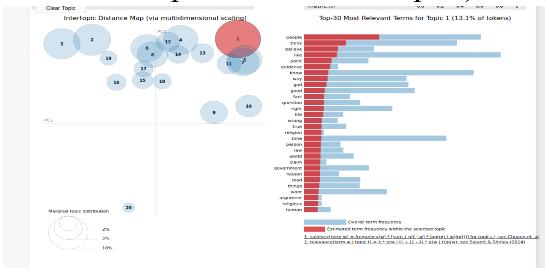
"TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents." This method was designed for document search and information retrieval. To find the TF-IDF value, we can just simply use a Gensim build-in method called tfidfModel.

Latent Dirichlet Allocation

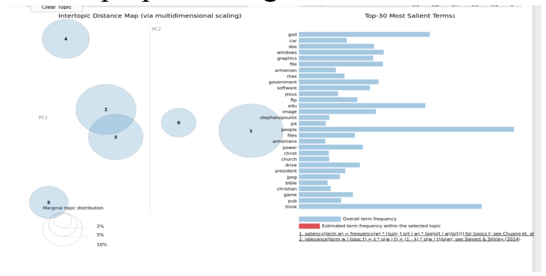
"LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar." We also used the Gensim build-in method to calculate the LDA model. The figure below is a LDA model visualization via pyLDAvis . Upon closer inspection, we found that we two topics are close to each other, they also contain similar or closely related words.



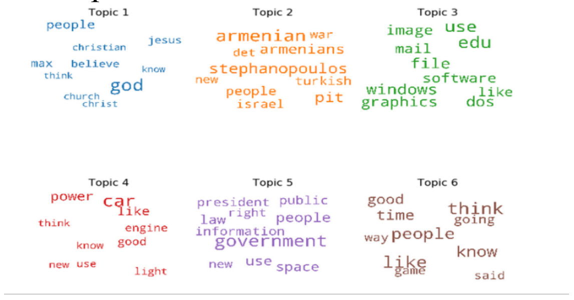
We also test the LDA visualization with different visluzation methods. A simpler version with different filters that we design by ourselves. It is quite obvious that we have worse results (all topics are super close to each other. This should not happened because some topics are vastly different compared with other topics)



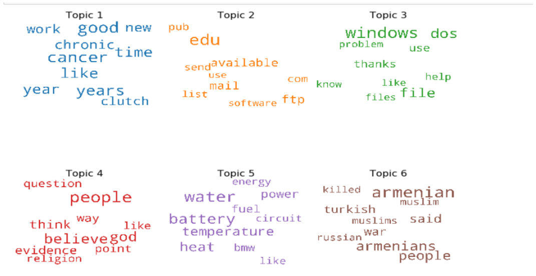
The figure below shows LDA visualization when the number of topics is 6. We can find out that we have 6 vastly different topics. We observed no significant differences between the built-in preprocessing method and our own preprocessing method.



The figure below is another LDA visualization method which shows the top 10 words for each topic.



When we use a simpler version of preprocessing, it is also pretty obvious that it generates worse results. We found nearly duplicate words (that should be filtered out). For example, in topic 6, it has both Armenian and Armenians.


Topic 1
work good new chronic time cancer like year years clutch
Topic 2
pub edu send available use com mail list software ftp
Topic 3
windows dos problem use thanks know like help files file
Topic 4
question people think way like believe god evidence point religion
Topic 5
energy water power fuel battery circuit temperature heat bmw like
Topic 6
killed armenian muslim turkish muslims said russian war armenians people

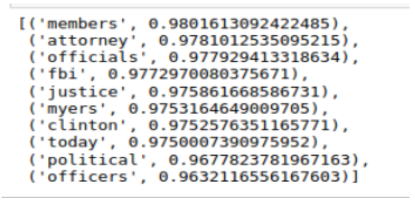
Conclusion about LDA

It is pretty clear that built-in preprocessing methods generate better results for two different topic numbers (6 and 20). However, the built-in preprocessing is not perfect and still needs improvement. Both methods produce a decent result when the number of topics is 6.

Word2Vec

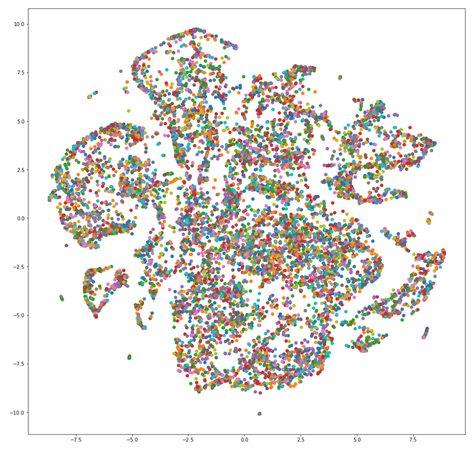
Word2Vec assumes that two words sharing similar contexts also share similar meanings and similar vector representations from the model. Therefore, Word2Vec is an excellent way to compute the similarity and relation between two different documents.

For example, if we use the word “president” as input, we will receive the following result. All those words are closely related to the word president. This proves that we get the correct Word2Vec document representation.



```
[('members', 0.9801613092422485),
 ('attorney', 0.9781012535095215),
 ('officials', 0.977929413318634),
 ('fbi', 0.9772970080375671),
 ('justice', 0.975861668586731),
 ('myers', 0.9753164649009705),
 ('clinton', 0.9752576351165771),
 ('today', 0.9750007390975952),
 ('political', 0.9677823781967163),
 ('officers', 0.9632116556167603)]
```

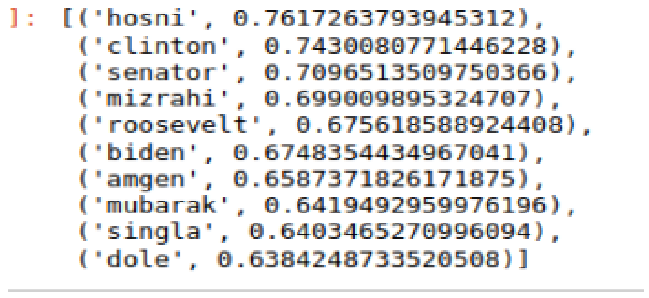
The figure below is the T-SNE graph for our Word2Vec document representation/



Doc2Vec

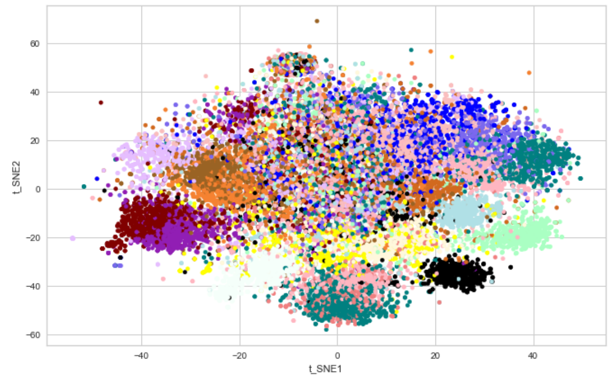
Doc2Vec is kind of similar to Word2Vec, but Doc2Vec still retains the context within the text. Therefore, we can expect that we will have better results compared with the Word2Vec method.

We also used the word "president" as an example for our Doc2Vec representation. Based on thg result, it is more closely related to the input word (Majority of words here are prominent political figures).



```
]:  [('hosni', 0.7617263793945312),
     ('clinton', 0.7430080771446228),
     ('senator', 0.7096513509750366),
     ('mizrahi', 0.699009895324707),
     ('roosevelt', 0.675618588924408),
     ('biden', 0.6748354434967041),
     ('amgen', 0.6587371826171875),
     ('mubarak', 0.6419492959976196),
     ('singla', 0.6403465270996094),
     ('dole', 0.6384248733520508)]
```

The result from the T-SNE graph also has massive improvement upon Word2Vec. Based on the observation solely, we can find out that it produces a much better clustering result.



Analysis the result

The table below shows the NMI analysis for different document representation methods. We can clearly notice that as document representation becomes more and more sophisticated, the NMI values are also getting higher and higher.

Bag-of-words (NMI Value)	**0.02373109453712898**
TF-IDF (NMI Value)	**0.08080094822289248**
Doc2Vec (NMI Value)	**0.314032989698696**

Figure below is the K-means clustering for Bag-of-Words document representation

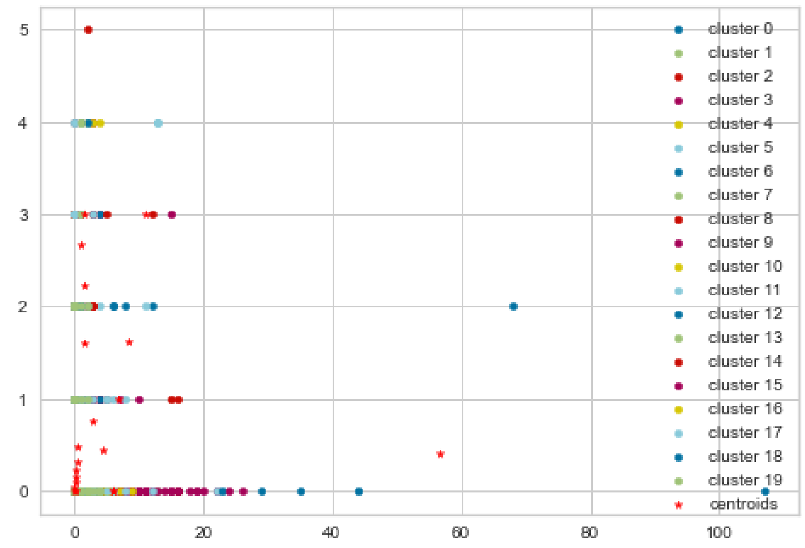


Figure below is the K-means clustering for TF-IDF document representation

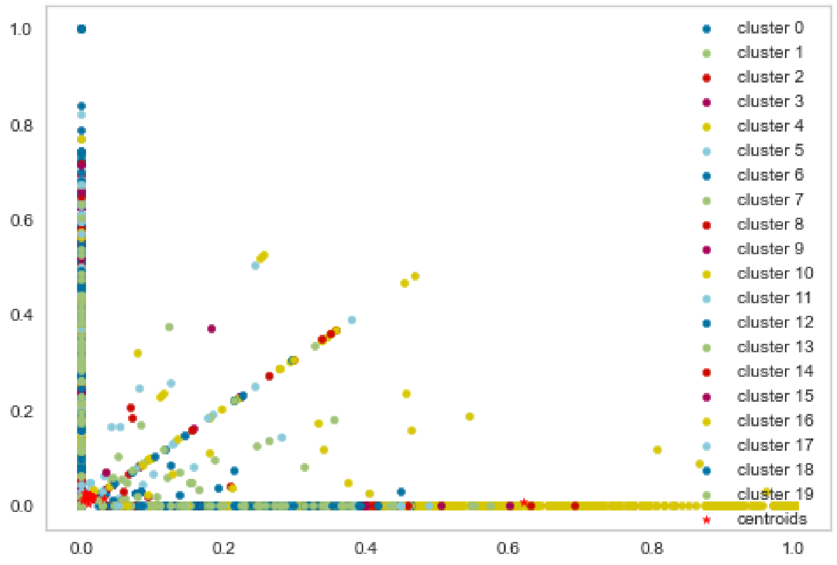
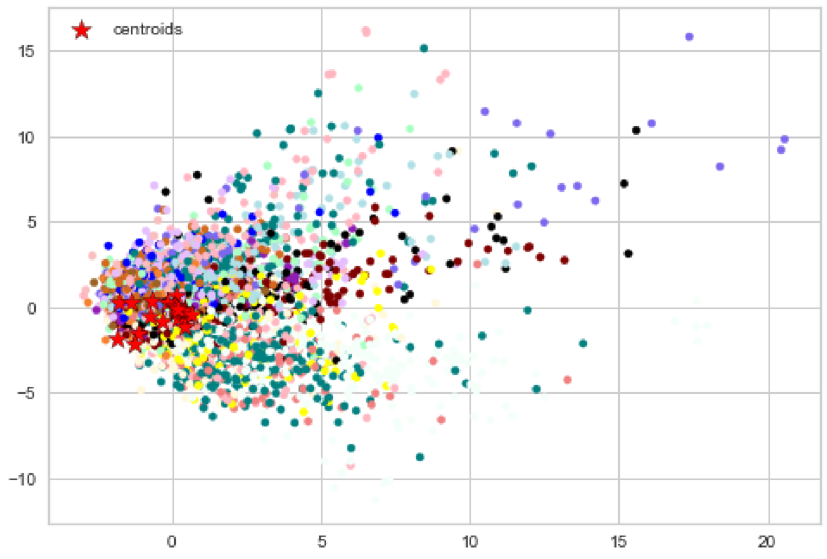


Figure below is the K-means clustering for Doc2Vec document representation

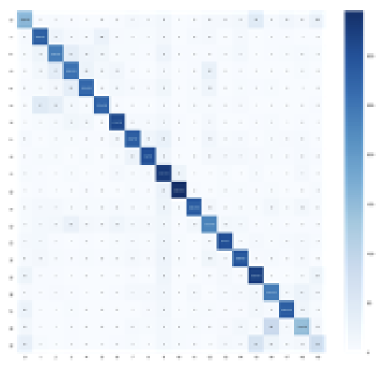


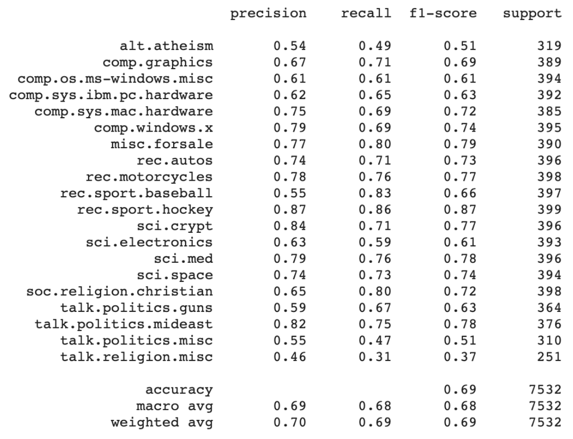
From the figures above we can observe that we are getting better results as document representation becomes more and more sophisticated. In addition, since we are using the K-means clustering method for Doc2Vec in this part, it also explains why it has worse results compared with the T-SNE clustering method.

Supervised Learning Method ---- Supported Vector Machines

For previous parts of the project, we only used unsupervised machine learning, which may not have the most accurate results. One widely used supervised machine learning algorithm Support Vector Machines (SVMs) offered very high accuracy compared to other machine learning algorithms.

We perform Support Vector Machines classification on tf-idf representation. The figure on the left below is the heatmap we generated by SVMs classification. The figure on the right below is the performance metrics in which we can check each topic's accuracy rate, recall rate, and f1-score in a single graph. The accuracy can be computed by comparing actual test set values and predicted values. As we can see we got a very good accuracy for each topic. The NMI value also has an obvious improvement compared with the K-means before.



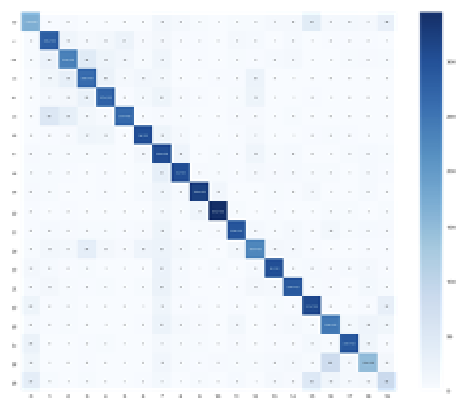


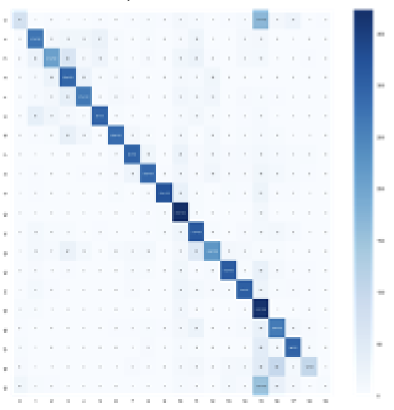
```
                          precision    recall  f1-score   support

             alt.atheism       0.54      0.49      0.51       319
           comp.graphics       0.67      0.71      0.69       389
 comp.os.ms-windows.misc       0.61      0.61      0.61       394
comp.sys.ibm.pc.hardware       0.62      0.65      0.63       392
   comp.sys.mac.hardware       0.75      0.69      0.72       385
          comp.windows.x       0.79      0.69      0.74       395
            misc.forsale       0.77      0.80      0.79       390
               rec.autos       0.74      0.71      0.73       396
         rec.motorcycles       0.78      0.76      0.77       398
      rec.sport.baseball       0.55      0.83      0.66       397
        rec.sport.hockey       0.87      0.86      0.87       399
               sci.crypt       0.84      0.71      0.77       396
         sci.electronics       0.63      0.59      0.61       393
                 sci.med       0.79      0.76      0.78       396
               sci.space       0.74      0.73      0.74       394
  soc.religion.christian       0.65      0.80      0.72       398
      talk.politics.guns       0.59      0.67      0.63       364
   talk.politics.mideast       0.82      0.75      0.78       376
      talk.politics.misc       0.55      0.47      0.51       310
      talk.religion.misc       0.46      0.31      0.37       251

                accuracy                           0.69      7532
               macro avg       0.69      0.68      0.68      7532
            weighted avg       0.70      0.69      0.69      7532
```

The figure on the left below is generated by RNNs classification. The figure on the right below is generated by Naïve Bayes classification. As we can see Naïve Bayes classification has the highest NMI value, which clearly improved classification performance (around 3% NMI value) compared with SVMs classification. However, SVMs classification has a better accuracy rate.





	NMI	Accuracy	Recall
SVMs	0.5575970027139495	0.6919808815719597	0.6919808815719597
RNNs	0.5694433967581202	0.6979553903345725	0.6979553903345725
Naïve bayes	0.586141559437364	0.6845459373340415	0.6845459373340415