Midterm Project: Document Visualization

EECE 5642

Group 5: Ryan Birke, Josh English, and Douglas Schonholtz

Introduction

We applied natural language processing (NLP) techniques on the “20 Newsgroups” dataset. We used an improved preprocessing method to clean the data and made two vocabularies - a full vocabulary and a subset vocabulary consisting of the 2000 most frequent words. We then learned Bow, TF-IDF, LDA, and Doc2Vec models for each vocabulary respectively. Finally, the results of each model were visualized and the performance of each model was evaluated with the NMI metric.

Methodology

The dataset was cleaned by removing stop words, lemmatization, and removing words with 2 or fewer characters. Statistics of the cleaned dataset were found and can be seen in detail in the “preprocess_stats” figure. The statistics of the dataset were then visualized with two different methods, these visualizations can be found in the “preprocess_count” and “preprocess_frequency” figures. Our first vocabulary consisted of the full set of cleaned data. A subset of the 2000 most common words were used to build the second vocabulary (2k vocab).

For BoW, sklearn’s CountVectorizer function allowed us to tokenize the data into a series of words and fit a model to the data. PCA was used to reduce the dimensionality of the vocabulary into 2 or 3 dimensions. Finally, k-means was used from sklearn’s clustering library to cluster or classify the documents into categories and build a 2 or 3 dimensional graph of the data (see “bow_2d” and “bow_3d” for the 2k clustering visualizations). In the 3D graph, the size of the data point corresponds to its value in the 3rd dimension, although it is often hard to distinguish individual points.

TF-IDF followed the same steps as obtaining BoW models, but after using CountVectorizer and before using PCA the BoW data was transformed into TF-IDF representations with sklearn’s TfidfTransformer. The dimensionality was reduced using PCA. K-means was then used to classify the documents into categories and build 2 or 3 dimensional graphs of the data (see “tfidf_2d” and “tfidf_3d” for the 2k clustering visualizations).

LDA was used to find the topic distribution of the two corpora. Sklearn and gensim were both used to generate an lda model. The sklearn model excluded too common and very uncommon features. Gensim relied on the preprocessed data and used LdaMulticore to train the model using multiple workers, thereby reducing training time. The model was given 20 topics to find and the number of iterations was set to 100, with a chunk size of 1000.

Doc2Vec learned the word and document embedding space using the two corpora. These two different corpora were visualized with k-means, T-SNE, and PCA, all of which can be found as figures labeled appropriately. To attempt to optimize this process and to generate optimal results the NMI of several different factors were compared while being varied. The vector size in Doc2Vec was varied from 50 to 150, the number of epochs was varied from 5 to 100, and the number of clusters for k-means was also varied despite knowing the true number of clusters was 20.

Results

To see the effects of preprocessing please see figures “tfidf_stopwords” and “tfidf_3_letter_words”. The “tfidf_stopwords” show the resulting plot after following the steps discussed in the TF-IDF methodology using the 2k vocab and visualizing the results in a 2 dimensional graph, but only filtering the dataset with stopwords included in the nltk corpus library it had an NMI of 0.29. The “tfidf_3_letter_words” shows the same visualization, but with our improved preprocessing approach. This approach included extra stopwords (please see source code for details), more k-means iterations, and limiting the dataset to words of 3 characters or more. As we can see, the second figure has much clearer clusters, this is reflected in its NMI score of 0.38 which is an improvement of almost 10%. Thus, our improved preprocessing method yielded considerably improved clustering performance.

BoW on the 2k vocab had an NMI of just 0.05 (figure “bow_2d” for clustering). This is lackluster performance, especially given that our preprocessing method produced good results.

TF-IDF on the 2k vocab had an impressive NMI of 0.38 (figure “tfidf_2d” for clustering). This is decent performance but leaves much to be desired for automated document classification.

The NMI for LDA topic clustering with the full corpus was 0.4303, and 0.3111 with 2k (see “lda_kmeans” and “lda_kmeans_2k”). The T-SNE visualizations show distinct boundaries between clusters, with definite improvements in the 2k vocab (see “tsne_lda” and “tsne_lda_2k”). PyLDAvis revealed topics were much more easily distinguished with th 2k vocab (see “pyLDAvis_”).

For Doc2Vec we tested various training methodologies on the two different corpora. The optimal result was found to be to use the 2k vocab, with 14 epochs, and a vector size of 100. This ultimately generated an NMI of .42. Using the other corpus and vocabulary with the same optimal parameters generated an NMI of .3487 which is good but not nearly as good as the other corpus.

Analysis

To understand the effect of the number of clusters on k-means performance, extensive testing and analysis were performed on the TF-IDF and doc2vec models using the 2k vocab. For TF-IDF, the number of clusters was set to 2 and doubled for each trial. The results of the TF-IDF model with 2 clusters can be seen in the “tfidf_k2” figure. This classification produced an NMI of 0.23, which is surprisingly accurate for binary classification. Next, the number of clusters was doubled to 4, which generated the “tfidf_k4” figure. This classification produced an NMI of 0.35, which is rapidly approaching our maximum performance of 0.38. Again, the number of clusters was doubled to 8, which generated the “tfidf_k8” figure. This classification produced an NMI of 0.38, which matches our maximum performance. Next, the number of clusters was doubled to 16, which generated the “tfidf_k16” figure. This classification produced an NMI of 0.38, matching the performance of 8 clusters. Finally, the number of clusters was doubled to 32, which can be seen in the “tfidf_k32” figure. This classification produced an NMI of 0.37, showing slight overfitting. Given that the actual number of clusters is 20, it makes sense that the 8 and 16 cluster results had the best performance. The 32 cluster results show both that too many clusters negatively affects classification performance and that NMI is not a perfect performance metric.

Doc2vec was tested with a varying number of clusters for k-means. It was unsurprisingly found that 20 clusters were the optimal number of clusters to be used on the dataset near the actual number of clusters. Similar, to what was found to TF-IDF. 20 clusters generated an nmi of .4328. The k-means results for 18 clusters and 22 clusters both had NMI’s of .41. This makes sense as too few or too many clusters resulted in over or underfitting. Using significantly more than the actual number of clusters still generated very high NMI scores. 30 clusters generated an NMI of .4446, 100 clusters, .4496. 200 clusters, .4559. This is because NMI looks at the purity of each cluster compared to the commonalities between clusters and entropy. The purity goes up faster than the entropy as the size of the clusters get smaller, showing us NMI is not a perfect measure of clustering accuracy.[6]

Further analysis and improvement was seen when using the correct number of epochs and using the smaller corpus. 14 epochs seemed to be the optimal number as overfitting was observed after more epochs than this was used. The nmi of 100 epochs was .22 nearly exactly half of the optimal rate at 14 epochs. The vector size for doc2vec was also varied and from 50 to 150 and we found that 100 was roughly the optimal number for vectors.

There were two available ways to train an LDA model - using sklearn or gensim. Both methodologies were tested, and it was found that gensim was significantly (3x) slower than sklearn with no apparent increase in accuracy.

Conclusions

Optimizing topic modeling still feels like more of an art than a science. We found that constant tinkering with our parameters for our preprocessing, BOW, TF-IDF, LDA and Doc2Vec was one of the best ways to see optimization. In the future, we are excited to learn about further ways of optimizing this process so that each decision for optimization, whether it is a specific stop word to remove or a number of epochs to run a doc2vec model on can ultimately be more objective and not based largely on doing binary searches on the parameters seeking to find optimal parameters and visualizations.

References

[1] “Colaboratory.” Colaboratory, Google, colab.research.google.com/.

[2] Community, Python. “Python.” Python, 3.5.7, Python, Python.org.

[3] “Gensim Tutorial - A Complete Beginners Guide – Machine Learning Plus.” Machine Learning Plus, 18 Oct. 2018, www.machinelearningplus.com/nlp/gensim-tutorial/?fbclid=IwAR1wbKmHRNgFO7kJbFYd3AQg1s0yLcWdljbAkv3I1VsOEWvQWDyIo1h49Cg.

[4] “Jupyter Notebooks.” Jupyter, 5.7.6, Jupyter.org.

[5] Nijessen, Rik, and Rik Nijessen. “Automatic Topic Clustering Using Doc2Vec.” Towards Data Science, Towards Data Science, 15 May 2017, towardsdatascience.com/automatic-topic-clustering-using-doc2vec-e1cea88449c.

[6] “Normalized Mutual Information.” https://course.ccs.neu.edu/cs6140sp15/7_locality_cluster/Assignment-6/NMI.pdf

[7] Rehurek, Radim. “Gensim.” Gensim, 3.7.1, radimrehurek.com/gensim/index.html. “Scikit-Learn.”

[8] Scikit-Learn, 0.20.3, INRIA, scikit-learn.org/stable/.