20 Newsgroup dataset analysis Topic modelling and Clustering

Sanket Purohit:	00001607190
Sneh Desai:	00001616217
Soroor Ghandelli:	00001605110

---

1 Introduction

Utilizing 20 Newsgroup dataset which, involves around 18000+ newsgroups posts on 20 themes, we perform preprocessing on it and imagine the term-recurrence dispersion. Then, at that point, vectorize these archives utilizing BoW, TF-IDF, LDA, Word2Vec and Doc2Vec models. We envision the point circulation for LDA model and word and archive installing space for Word2Vec and Doc2Vec models. Once, we get vector portrayal of the archives, we performed bunching utilizing Kmeans grouping model. One regulated arrangement model is utilized to characterize the archives into 20 distinct classes.

2 Approach

2.1 Preprocessing

The preprocessing steps incorporate capitalized letters into lowercase letters, stopword removal, accentuation mark expulsion, digits removal, and guaranteeing word length is more than 3. We then, at that point, utilize further developed preprocessing like n-grams and lemmatization. We likewise eliminate headers, footers and statements from the records. To confirm our preprocessed information we utilize the world cloud bundle to a visualize the relative multitude of normal words in the model. This likewise assists us with checking if any longer preprocessing is expected before we train the model.

2.2 Build Vocabulary

We now build different vocabularies using two different filtering techniques:

1. Term frequency filtering
2. Document frequency filtering

A combination of these filtering techniques and above-mentioned preprocessing methods are used to obtain 3 different dictionaries. We will use these 3 dictionaries and observe how it impacts our text classification results.

i) `vocab_v1`- combination of bigram, lemmatization, and term frequency filtering($tf > 10$)(main vocab)

ii) `vocab_v2`- combination of bigram, lemmatization, document frequency filtering ($df > 5$), and 2000 most frequent words.

iii) `vocab3`- Combination of bigram,trigram, lemmatization, and term frequency filter ($tf > 10$)

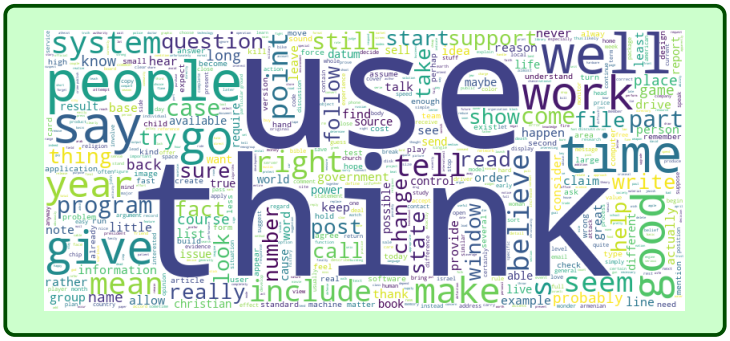


Figure 1: Caption

2.3 Bag-of-words (BoW) and TF-IDF model with `vocab_v1`

2.3.1 Bag-Of-Words (BoW):

For Bag-Of-Words model, we have used two different libararies: gensim and sklearn. To avoid high space complexity and high computational requirement of gensim MatrixSimilarity function, we have used sklearn CountVectorizer to create sparse matrix for BoW method using `Vocab_v1`. Gensim library is used for building BoW using 2k top frequency words. Using gensim library, we used the tokenized word list to create a dictionary. Each term is mapped to a numeric id. Now, using doc2bow function, we created bag of words representation for our Vocab list. We converted this to sparse matrix form which can be manipulated just like the dense version of a matrix. Then using matrix similarity, we computed the similarities between each document and clustered them into a group of 20 clusters using Kmeans model. NMI score for kmeans clustering using `Vocab_v1` and `Vocab_v2` is 0.24034 and 0.0586 respectively.

2.3.2 TF-IDF

The bag-of-words(integer values) corpus obtained from `Vocab_v1` is used to train a TF-IDF model. This model takes an integer-valued vector, performs transformations, and returns a real-valued vector of the same dimensionality. The features which are rare in the corpus will have their values increased. The model results are then converted into a dense matrix using gensim.matutils. This matrix becomes the input for our K-means clustering model. We then run our K-means model to obtain the clustering results. The K-means results have an NMI score of 0.2988. We also ran the clustering on TF-IDF representation obtained from `Vocab_v2`. And the NMI score of clustering with this representation is 0.183.

2.3.3 LDA

LDA is a transformation from bag-of-word counts into a topic space of lower dimensionality. Initially, we trained the base LDA model using `Vocab_v1` and topic count as 20. With this model, we obtain a coherence score of 0.5547. We train the model with the above-selected values to get topic representations for the documents in the corpus. Further, the `Lda_matrix` obtained from this model becomes the input matrix for K-means clustering. This time,with LDA topic representation, we obtain an NMI score of 0.33688

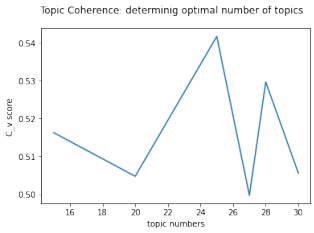


Figure 2: Caption

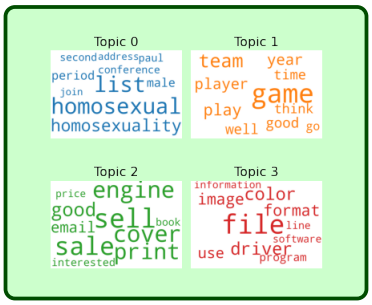


Figure 3: Caption

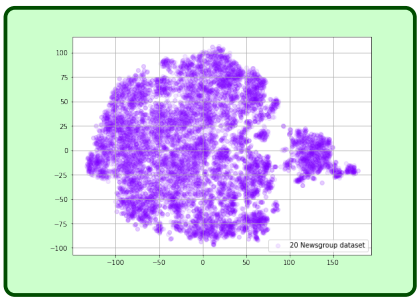
2.4 Word2Vec and Doc2Vec models on `Vocab_v1`

2.4.1 Word2Vec using `Vocab_v1`

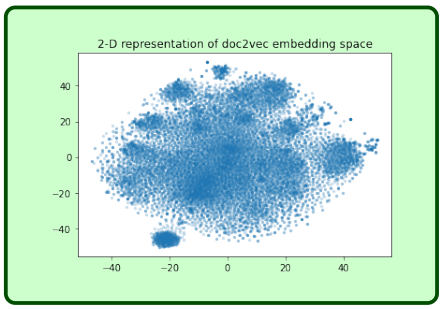
Using gensim's word2vec library, we created a model with 100 hidden layers and a minimum word count of 40. We trained this model over `Vocab_v1` and then obtained the Word Vectors for each word in the vocabulary. Using these word vectors, we performed Kmeans clustering over it and extracted centroid for each cluster. Obtained words in each cluster that was closest to the cluster center. We created a word cloud displaying words of a cluster for each of the 20 clusters.

2.4.2 Doc2Vec using `Vocab_v1`

Using gensim's TaggedDocument library, we prepared the training data for the Doc2Vec model. Then using this data, we trained the Doc2Vec model with a vector size of 50 and alpha value of 0.025. The resultant document vectors were used for clustering using kmeans clustering model. We saw the best clustering results for this method.Visualized the learned word embedding space for Word2Vec and document embedding space for Doc2Vec using t-SNE1.



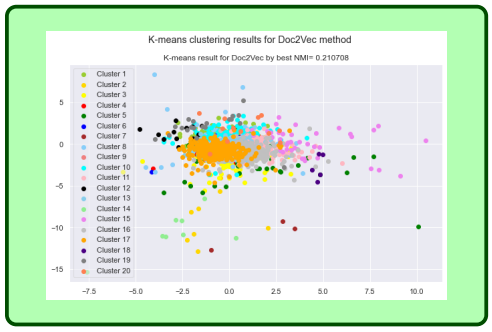
(a) 20 News Group Dataset



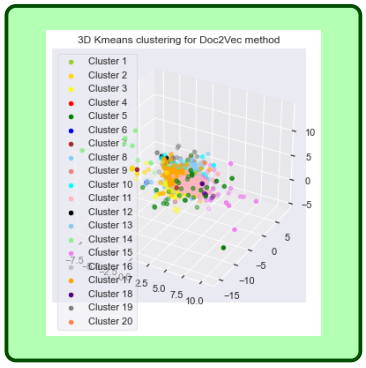
(b) 2-D representation of doc2vec embedding space

2.5 K-Means

We performed clustering using kmeans model for different document representations. Clustering results for all these methods is listed in this Section . We achieved the best NMI score for the Doc2Vec representation method which is 0.4124. Following figure shows the scatter plot for Kmeans clustering results for 20 Newgroup dataset. We created a new vocabulary `Vocab_v2` containing top 2k high frequency words. Then, we repeated the entire process for each document representation type using `Vocab_v2`. NMI score for each document representation method is given in Section.



(c) Kmeans



(d) 3-D KMeans

3 Methodology

3.1 Data preprocessing and Vocabulary building:

For data preprocessing and analysis we have mainly used numpy, pandas, matplotlib and gensim library functions. For transforming the texts documents to document vector form, we have used gensim and sklearn libraries. Different methods used to create document vector representations include: BoW, TF-IDF, LDA, Word2Vec, and Doc2Vec.
Training Data Shape (13192,15)
Testing Data Shape (5654,15)
Recommended Parameters after Tuning : ('C': 10, 'gamma': 0.01, 'kernel': 'rbf')

3.2 Clustering model development:

To transform the document into vector form, we have primarily used gensim and sklearn library functions. Different clustering models used for comparing the clustering performance include: Kmeans, GaussianMixture, MiniBatchKmeans, and Fuzzy clustering model.

4 Perfermance Metrics

For the evaluation of the performance of our clustering models, we have used `normalized_mutual_info_score` which gives the normalized mutual information between two clustering. It is an external index metric for evaluating clustering solutions. The highest NMI value that we achieved is 0.4124 for the Doc2Vec document representation method.

5 Clustering Results

The table below represents the NMI score for Kmeans clustering using different document representation for the 20 NewsGroup dataset.

	BoW	TF-IDF	LDA	Doc2Vec
Vocabulary 1	0.240	0.18386	0.31507	0.194
Vocabulary 2	0.0587	0.0813	0.2564	0.3795

The table below represents the NMI score for different clustering models used for clustering the 20 NewsGroup Dataset.

	K-MEANS	GAUSSIAN-MIX	FUZZY-CLUSTERING	MINIBATCH-KMEANS
Vocabulary 1	0.24034	0.1325	0.013374	0.2850

6 Supervised Classification using SVM

6.1 Motivation and Approach

The Internet has become the prime source of information and all sort of news nowadays. The amount of data on the Internet is rapidly increasing. It makes it difficult for traditional analytics methods to analyze and visualize these data. To address these shortcomings of traditional methods, we have used the Machine learning classification model. It classifies the 20NewsGroup dataset into 20 different classes. We decided to use SVM classifier as this model is good for classifying both linear as well as nonlinear data. First, we split the Doc2Vec document representation into training and test dataset. Then, using optimal parameter values, train the SVM model and use it to predict the class labels for test data. Also, perform cross validation to validate the performance of the model.

6.2 Methodology

Sklearn's `train_test_split` library is used to divide the dataset into training and test parts. Then we tuned the SVM hyperparameters on the training dataset using the Grid search approach. Using these recommended hyperparameter values, we performed model fitting on the 20NewsGroup training dataset. After this, we used stratified K-fold cross-validation to evaluate the performance of the model. Using this SVM model, predicted the class labels for test data and calculated its performance metrics.

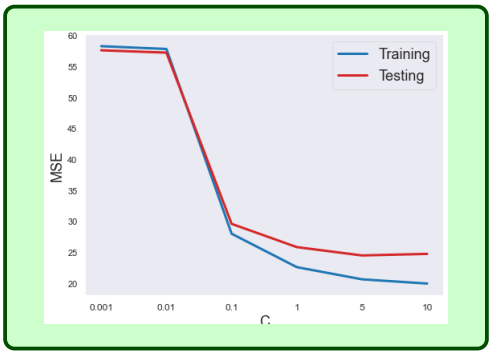


Figure 4: Caption

6.3 Performance Metrics

Accuracy, precision, recall, and F1-score are used to evaluate the performance of the model. We achieved an accuracy of .6178 for our dataset using SVM classification model.

6.4 Results

Table below represents the accuracy, precision, recall, and F1-score for the 20 NewsGroup dataset classification using SVM classification model.

	Accuracy	Precision	Recall	F1 Score
SVM	0.525	0.55	0.55	0.5348

6.5 Conclusion

After performing the classification task on the 20NewsGroup dataset, we found that the SVM model performed well on classifying our dataset into 20 different

classes. We achieved an accuracy of 0.6178 and an F1-score of 0.6142 using the SVM model

7 References

1. https://towardsdatascience.com/implementing-multi-class-text-classification-with-doc2vec-df7c3812824d
2. https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-earn/
3. https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d
4. https://radimrehurek.com/gensim/autoexamples/core/runcorporaandvectorspaces.html#sphx-glr-auto-examples-core-run-corpora-and-vector-spaces-py
5. https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/
6. https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
7. https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/
8. https://radimrehurek.com/gensim/autoexamples/core/runtopicsandtransformations. html#sphx-glr-auto-examples-core-run-topics-and-transformations-py
9. https://radimrehurek.com/gensim/autoexamples/tutorials/runlda.html#sphx-glr-auto-examples-tutorials-run-lda-py
10. https://radimrehurek.com/gensim/autoexamples/tutorials/runword2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py
11. https://radimrehurek.com/gensim/autoexamples/tutorials/rundoc2veclee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py