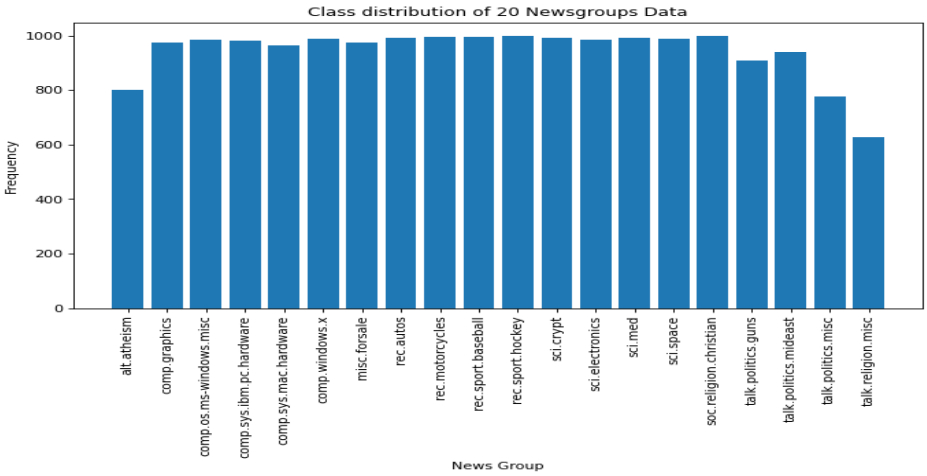
Name: 1) Sameeksha Raj Shimoga Basavaraja (W1588408)
2) Manisha Singh (W1551133)

Title
20 Newsgroup dataset analysis, Topic Modelling and Clustering

Executive Summary:
Objective of this program is to preprocess and build vocabulary for 20 Newsgroup dataset using BoW, TF-IDF, LDA, Word2Vec and Doc2Vec models. Using this vocabulary, data points with similar behavior are merged into one group called clusters. Then, perform comparative analysis of document clustering using Kmeans clustering model.

1. Introduction
Using 20 Newsgroup dataset which, comprises around 18000+ newsgroups posts on 20 topics, we perform preprocessing on it and visualize the term-frequency distribution. Then, vectorize these documents using BoW, TF-IDF, LDA, Word2Vec and Doc2Vec models. We visualize the topic distribution for LDA model and word and document embedding space for Word2Vec and Doc2Vec models. Once, we get vector representation of the documents, we performed clustering using Kmeans clustering model. One supervised classification model is used to classify the documents into 20 different classes.

2. Approach – Vocab_v1
2.1. Preprocess the 20 Newsgroup dataset and visualize its statistical information
Cleaning
Before we get into text classification, we will perform cleaning and pre-processing of our dataset. In our cleaning we discard sections from each document like headers, footers, and quotes. We then use this cleaned data and observe its class distribution. The following figure describes the class distribution of our dataset.

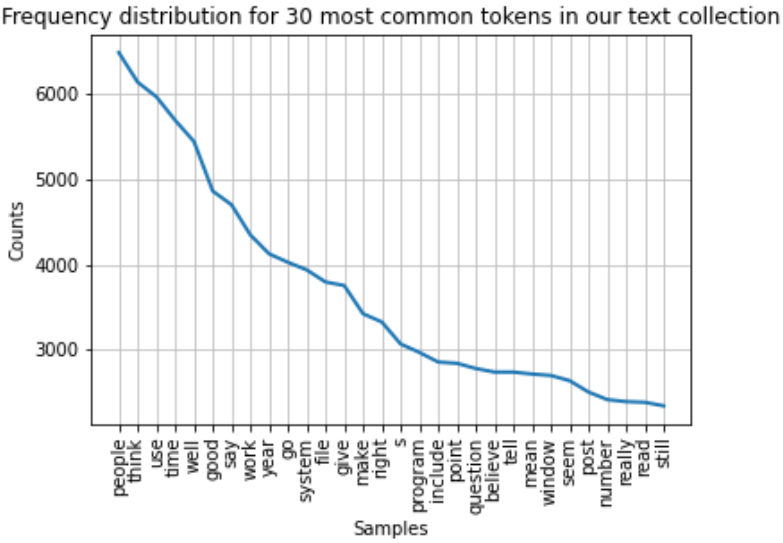


We see that the documents are evenly distributed among 20 topics.

Preprocessing
Our basic pre-processing steps include conversion of uppercase letters into lowercase letters, stopword removal, punctuation mark removal, digits removal, and word length greater than 3. We then use more advanced preprocessing like- n-grams and lemmatization

Exploratory Analysis on Preprocessed Data
To verify whether the preprocessing is good enough, we built a word cloud using the wordcloud package to get a visual representation of the most common words. This acts as a good check to see if more preprocessing is necessary before training the model.

Additionally, we plot the frequency distribution of tokens(words) to observe the 30 most common tokens in our text collection.



Build Vocabulary

We now build different vocabularies using two different filtering techniques

- term frequency filtering
- 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.

1) vocab_v1- combination of bigram, lemmatization, and term frequency filtering(tf >10)(main vocab)
2) vocab_v2- combination of bigram, lemmatization, document frequency filtering(df>5), and **2000 most frequent words**
3) vocab3- combination of bigram, trigram, lemmatization, and term frequency filtering(tf >10)

2.2. Bag-of-words (BoW) and TF-IDF model with Vocab_v1

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 0.1170 and 0.0586 respectively.

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 0.0864**.**

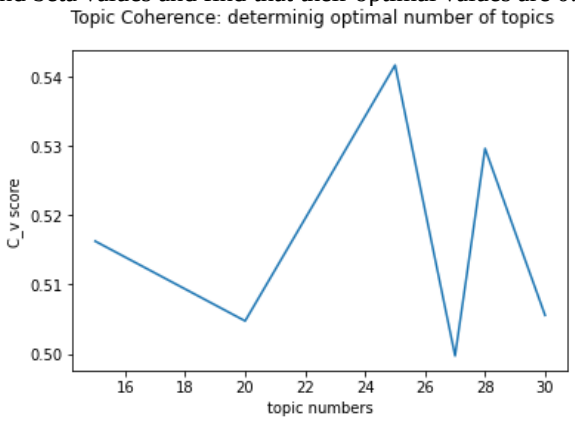
2.3. LDA model with Vocab v1

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 0.5047**.**

Experiment Analysis

Impact of Different topic numbers : Hyperparameter Tuning

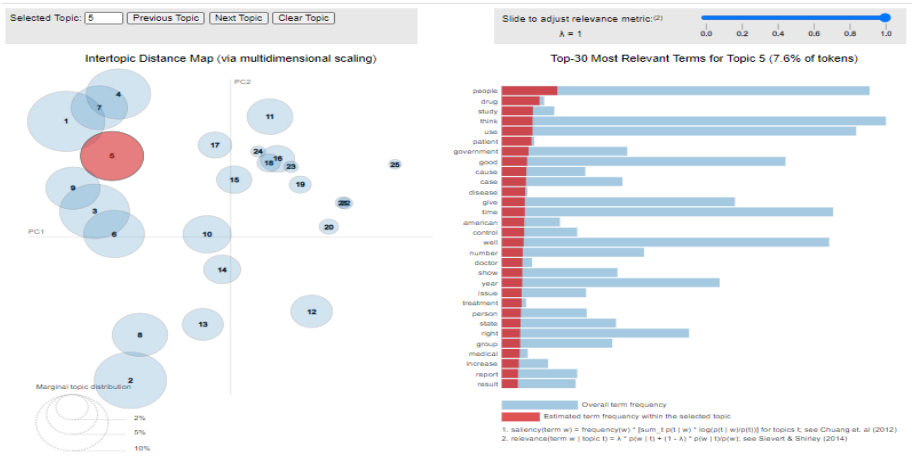
We move to tune the hyperparameters of our LDA model: changing topic numbers, alpha, and beta values. We determine the optimal number of topics by changing our topic number from 15 to 30 and observing the coherence values. The chart below highlights that the optimal topic count is 25 with a coherence score of 0.54167. We do a similar experiment for alpha and beta values and find that their optimal values are 0.01 and 0.9 respectively.



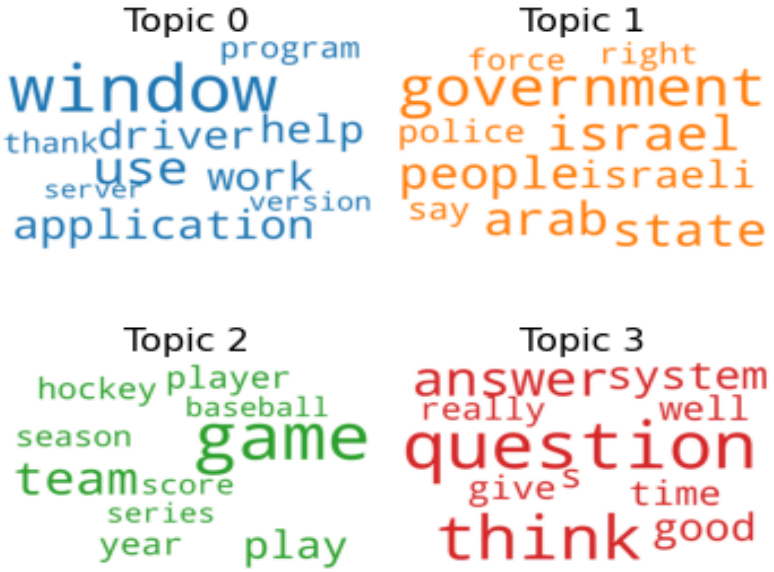
We train the model with the above-selected values to get topic representations for the documents in the corpus. Further, the lda_matix obtained from this model becomes the input matrix for K-means clustering. This time,with LDA topic representation, we obtain an NMI score of 0.33688

Visualizing the LDA topics

1) Topic visualization is done using pyLDAvis.

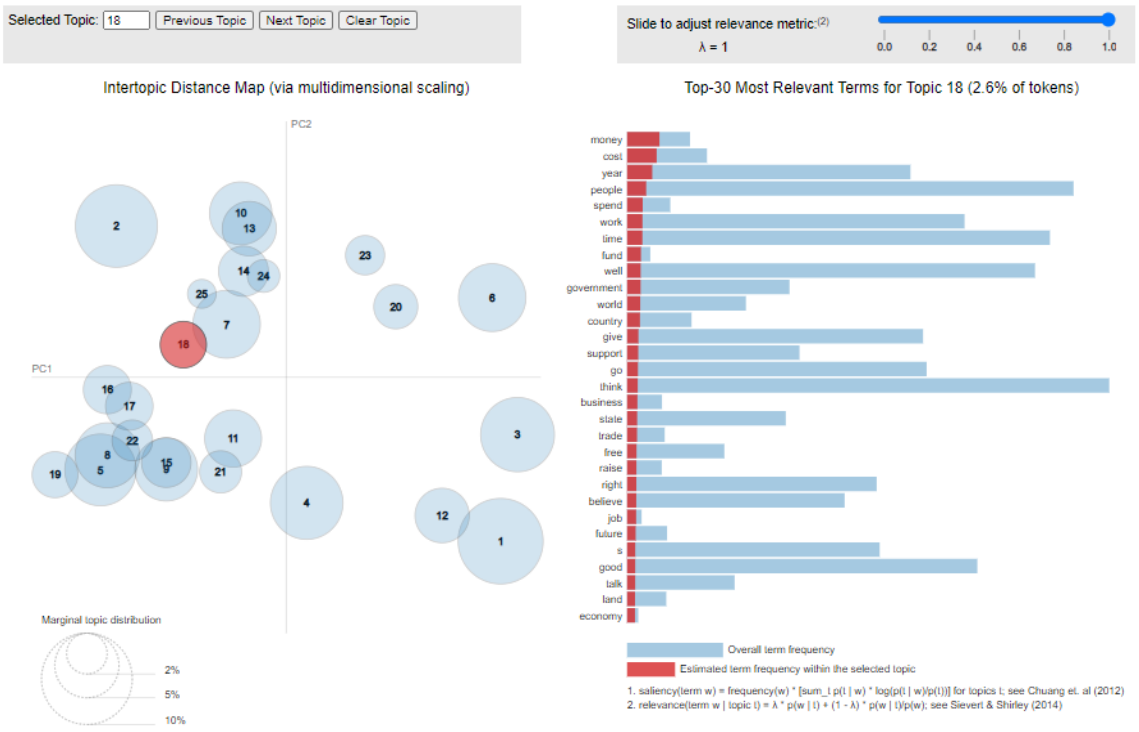


2) We can also visualize the top keywords in each topic with the help of word clouds. 4 topics and their top keywords are visualized below.



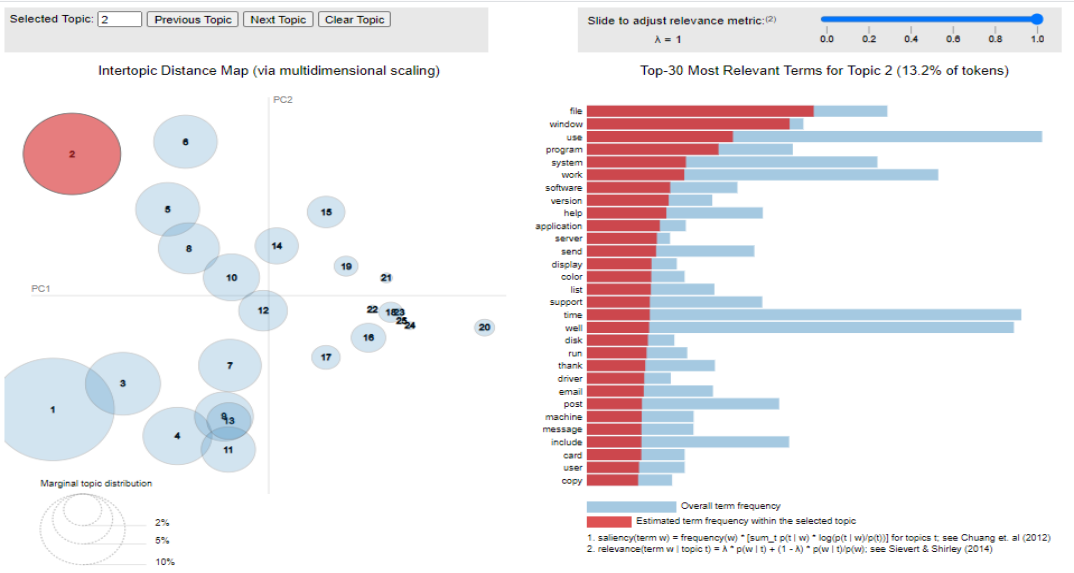
Impact of different preprocessing ways
In this section we compare the different LDA representations obtained by running the model using different vocabularies that were created in the initial sections of this project.

LDA modeling with 2k vocab (vocab_v2)
LDA coherence value is: **0.5511** (with topic count=25)
Clustering NMI score: **0.2975**



LDA visualization(vocab_v3)

LDA coherence value is: **0.5349** (with topic count=25)



Observations

When we look at the LDA topics obtained from using vocab_v2(top 2000 tokens), we notice that the topics are overlapping. Whereas, the topic representation obtained from using vocab_v3, which uses trigrams, results in better clusters with less overlap. Therefore, we can say that for this dataset (20 Newsgroups), performing a trigram to obtain the dictionary results in good topic modelling.

2.4. Word2Vec and Doc2Vec models upon Vocab v1

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.

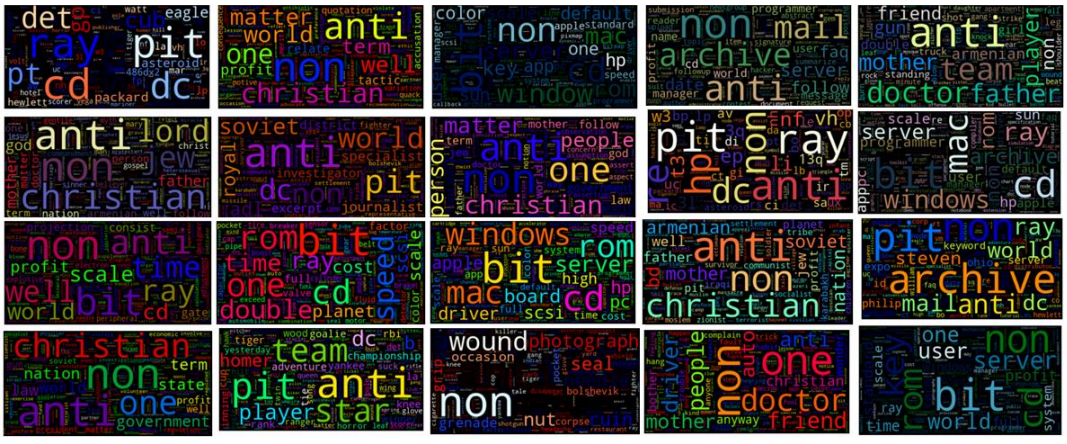


Figure: Word cloud showing words of 20 different clusters

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.

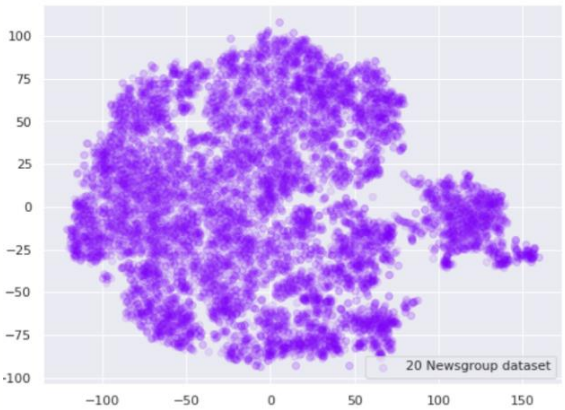


Figure: Learned word embedding space for word2vec

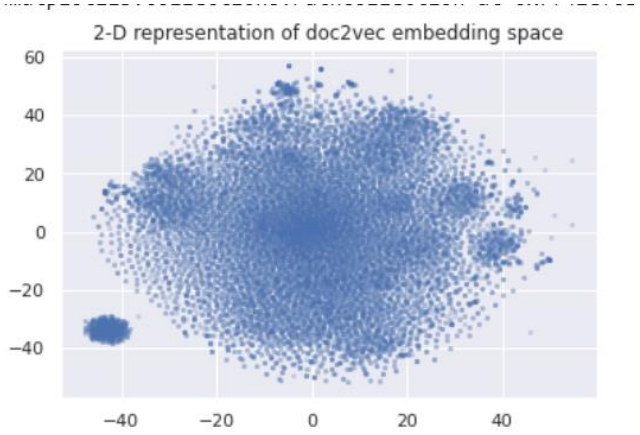


Figure: 2-D Doc2Vec embedding space

2.5. Document clustering by K-means

We performed clustering using kmeans model for different document representations. Clustering results for all these methods is listed in **Section 5.** 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.

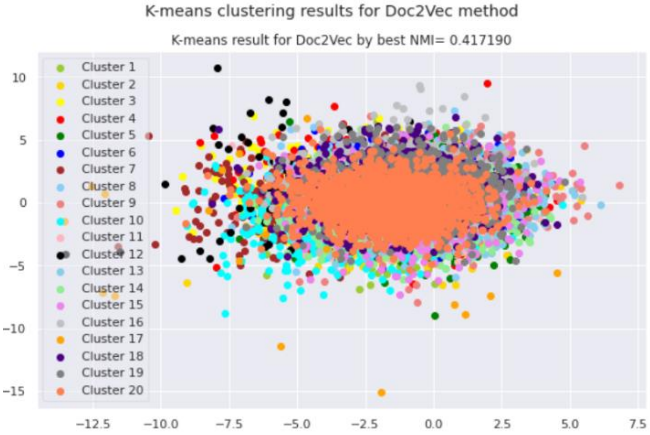


Figure: Kmeans clustering for 20 NewsGroup dataset clusters

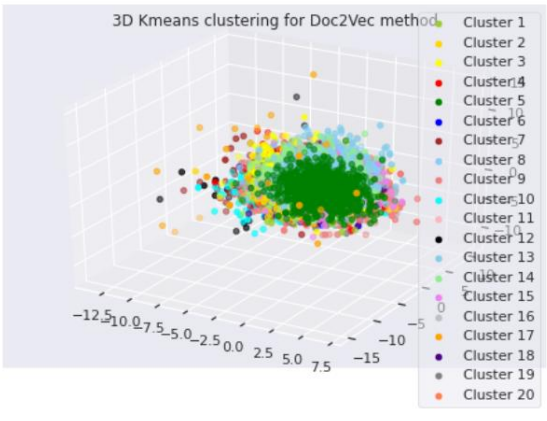


Figure: 3-D representation of 20 NewsGroup clusters

We created a new vocabulary Vocab_2 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 5.**

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.

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. Performance 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

Table 1 represents the NMI score for Kmeans clustering using different document representation for the 20 NewsGroup dataset.

	Normalized Mutual Information Score (NMI)			
	BoW	**TF-IDF**	**LDA**	**Doc2Vec**
Vocabulary 1	0.1170	0.2988	0.34607	0. 4124
Vocabulary 2	0.0586	0.0864	0.2975	0.3795

Table 1: K-means clustering result by NMI for each doc

Table 2 represents the NMI score for different clustering models used for clustering the 20 NewsGroup Dataset.

	Normalized Mutual Information Score (NMI)			
	Kmeans	**GaussianMixture**	**Fuzzy Clustering**	**MiniBatchKmeans**
Vocabulary 1 (Doc2Vec)	0.4124	0.2848	0.0074	0.2850

Table 2: NMI score for different clustering models

6. Supervised Classification using SVM:

6.1. Motivation: 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.

6.2. Approach: We decided to use SVM classifier as this model is good for classifying both linear as well as non-linear 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.

6.3. 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.

```
Training dataset shape:(13192, 50)
Test dataset shape:(5654, 50)
```

```
Recommended parameters after tuning: {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
```

Figure 3 shows training and testing loss over different values of C.

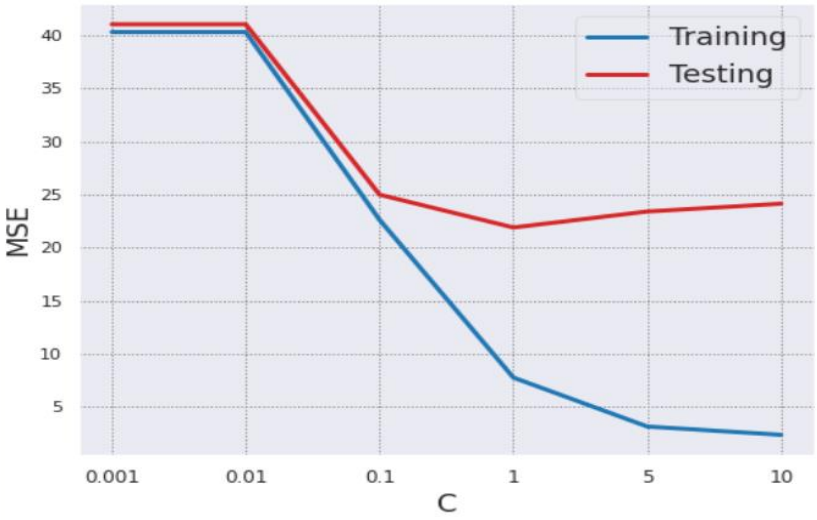


Figure: Training and test loss

6.4. 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.

6.5. Results:
Table 3 represents the accuracy, precision, recall, and F1-score for the 20 NewsGroup dataset classification using SVM classification model.

	Accuracy	**Precision**	**Recall**	**F1-score**
SVM model	0.6178	0.6200	0.6200	0.6142

Table 3: Performance metrics for SVM classification model

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

https://towardsdatascience.com/implementing-multi-class-text-classification-with-doc2vec-df7c3812824d
https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/
https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d
https://radimrehurek.com/gensim/auto_examples/core/run_corpora_and_vector_spaces.html#sphx-glr-auto-examples-core-run-corpora-and-vector-spaces-py
https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/
https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/