



20 Newsgroups Text Classification

AML-2304 Natural Language Processing

Presented by:

Kishan Lakhatariya (C0849341)

Harshkumar Patel (C0848343)

Mohammaddin Masbi (C0850500)

Kanan Trivedi (C0851492)

Professor: Harriet Huang

Contents

- Abstract
- Dataset
- Exploratory Data Analysis (EDA)
- Data Preprocessing
- Tokenization
- Stemming
- Train and Test
- Model Training
- Classification Report
- Confusion Matrix

Abstract

- Analyzed and processed text data of 20 newsgroups text datasets with different techniques to get meaningful words from the dataset and converted them into word vectors using two different word embedding strategies, TF-IDF and Word2Vec, and trained the vectorized data using three different machine learning models, from all the training we got maximum training accuracy of 96% and testing accuracy of 71% using Support Vector Machine Classifier with the vectorizer TF-IDF.
- It is essential to classify the new topics for the chatbots and AI assistants to give recommendations based on the reader's interest in addressing this problem; we have chosen the news dataset with 20 different topics as labels and trained the model to predict the news text.

```
1 # Mapping labels with its integer values (Label Encoding)
2 lable_map = dict((i, j) for i, j in enumerate(news_dataset.target_names))
3 lable_map

{0: 'alt.atheism',
 1: 'comp.graphics',
 2: 'comp.os.ms-windows.misc',
 3: 'comp.sys.ibm.pc.hardware',
 4: 'comp.sys.mac.hardware',
 5: 'comp.windows.x',
 6: 'misc.forsale',
 7: 'rec.autos',
 8: 'rec.motorcycles',
 9: 'rec.sport.baseball',
10: 'rec.sport.hockey',
11: 'sci.crypt',
12: 'sci.electronics',
13: 'sci.med',
14: 'sci.space',
15: 'soc.religion.christian',
16: 'talk.politics.guns',
17: 'talk.politics.mideast',
18: 'talk.politics.misc',
19: 'talk.religion.misc'}
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18846 entries, 0 to 18845
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   data        18846 non-null   object 
 1   target      18846 non-null   int64  
 2   target_label 18846 non-null   object 
dtypes: int64(1), object(2)
memory usage: 441.8+ KB
```

```
1 # Converting train dataset into pandas dataframe and storing into train_df variable
2 df = pd.DataFrame({'data': news_dataset.data, 'target': news_dataset.target})
3 df["target_label"] = df["target"].map(lable_map)
4 df.head()

      data  target  target_label
0 \n\nI am sure some bashers of Pens fans are pr...    10      rec.sport.hockey
1 My brother is in the market for a high-perform...     3  comp.sys.ibm.pc.hardware
2 \n\n\n\n\tFinally you said what you dream abou...    17  talk.politics.mideast
3 \nThink!\n\nIt's the SCSI card doing the DMA t...     3  comp.sys.ibm.pc.hardware
4 1) I have an old Jasmine drive which I cann...     4      comp.sys.mac.hardware
```

Dataset

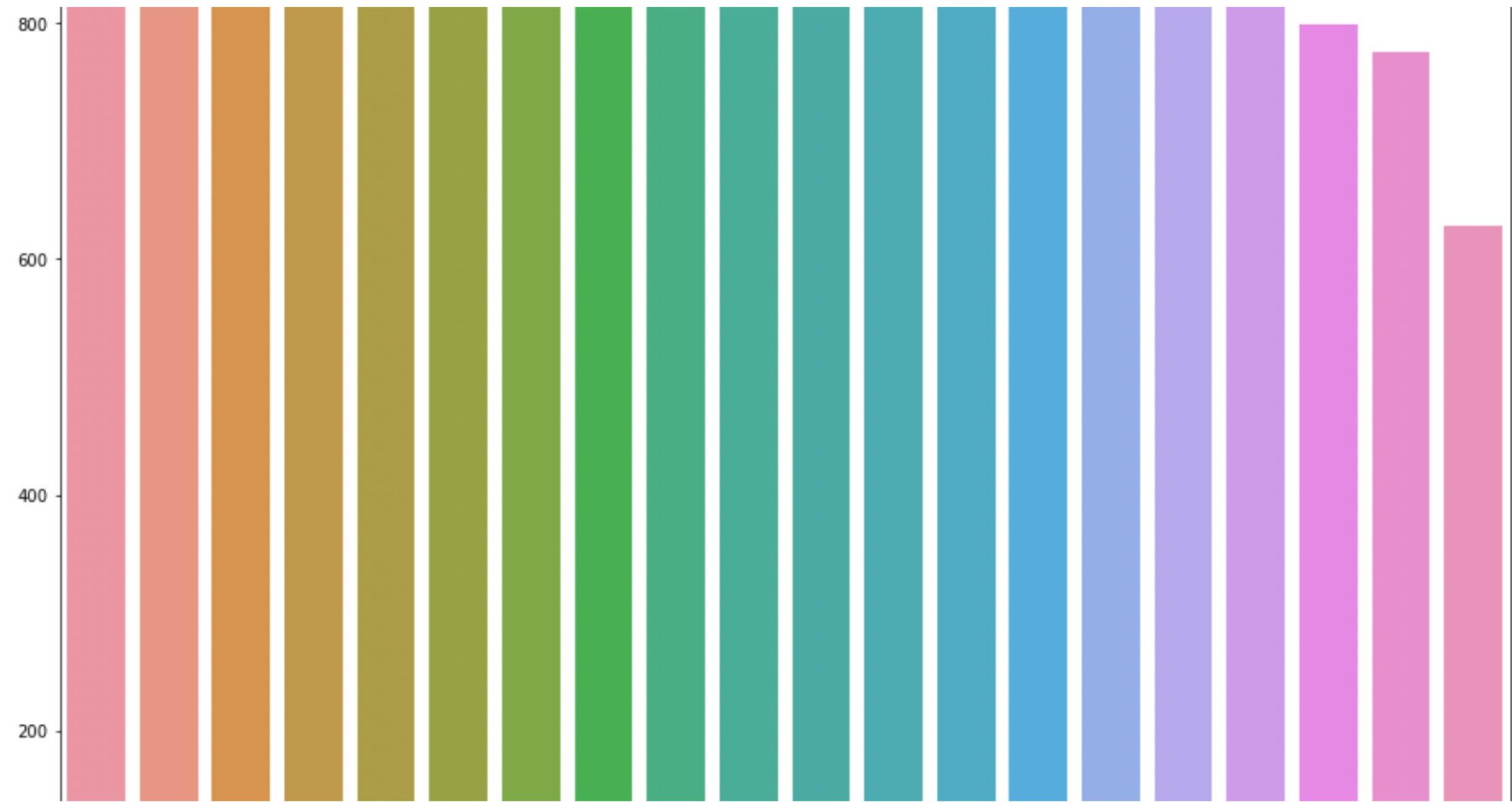
- We have used the `fetch_20newsgroups` API to get the dataset which contains 18846 samples with 2 columns of data and a target.
- As the target is the integer value, we have mapped the target_labels with it for further analysis
- The initial shape of the data is 18846 rows and 3 columns.

Exploratory Data Analysis (EDA)

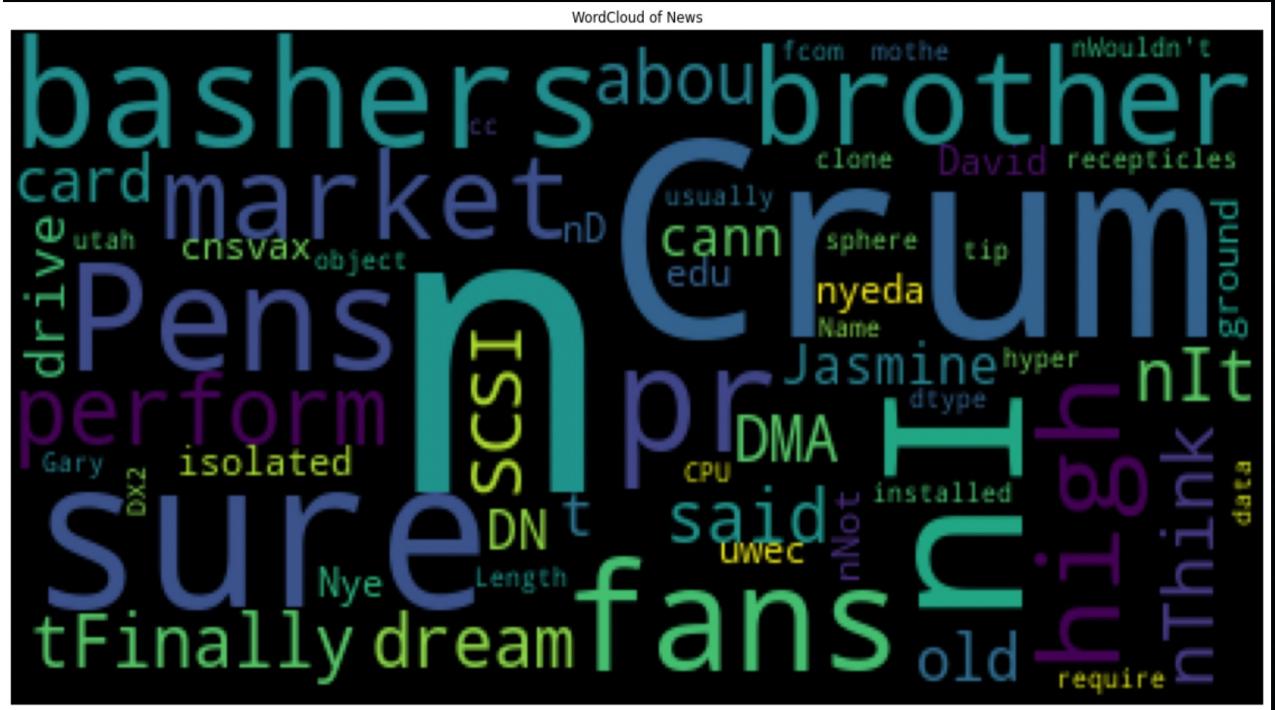
- We have three data initially
- i. Data: which contains the text of the news.
- ii. Target: which is the integer value of labels.
- iii. Target_labels: which is the string of the labels.
- The total count and frequency of the labels average are 750+.

```
1 # Getting the exact count of Labels
2 df.target_label.value_counts()
```

rec.sport.hockey	999
soc.religion.christian	997
rec.motorcycles	996
rec.sport.baseball	994
sci.crypt	991
rec.autos	990
sci.med	990
comp.windows.x	988
sci.space	987
comp.os.ms-windows.misc	985
sci.electronics	984
comp.sys.ibm.pc.hardware	982
misc.forsale	975
comp.graphics	973
comp.sys.mac.hardware	963
talk.politics.mideast	940
talk.politics.guns	910
alt.atheism	799
talk.politics.misc	775
talk.religion.misc	628
Name: target_label, dtype: int64	



```
1 # plot word count for WordCloud  
2 wc = WordCloud().WordCloud()  
3 wc.generate(str(df['data'][1]))  
4 fig = plt.figure(figsize=[20, 22])  
5 plt.title('WordCloud of News')  
6 plt.axis('off')  
7 plt.imshow(wc)  
8 plt.show()
```



Pre-processing

- We performed pre-processing steps to convert text data into meaningful data to make the computer understandable the text data.
 - First, we plot the word cloud to know the frequency of the words, and we discover that many un-useful words have higher frequencies than essential words.
 - Here we can see that the occurrence of the “n” is higher in the text data.

```

1 # Converting text paragraph into tokens using nltk.word_tokenize function and also removing special characters and un-usefull text
2 # And returning tokens list
3 def tokenize(data):
4     tokens = nltk.word_tokenize(data)
5     return [word for word in tokens if word.isalpha()]

1 # Applying tokenize function to data column of dataset and storing results into column name tokenized
2 df['tokenized'] = df.apply(lambda x: tokenize(x['data']), axis=1)
3 df.head()

    data   target   target_label
0 \n\nI am sure some bashers of Pens fans are pr...    10      rec.sport.hockey [I, am, sure, some, bashers, of, Pens, fans, a...
1 My brother is in the market for a high-perform...     3      comp.sys.ibm.pc.hardware [My, brother, is, in, the, market, for, a, vid...
2 \n\n\n\n\tFinally you said what you dream abou...    17      talk.politics.mideast [Finally, you, said, what, you, dream, about, ...
3 \nThink!\n\nIt's the SCSI card doing the DMA t...     3      comp.sys.ibm.pc.hardware [Think, It, the, SCSI, card, doing, the, DMA, ...
4 1) I have an old Jasmine drive which I cann...     4      comp.sys.mac.hardware [I, have, an, old, Jasmine, drive, which, I, c...

```

So, we must perform data cleaning, including removing symbols and ASCII characters, stopping word removal, stemming and lemmatizing.

We tokenized the data using NLTK's word_tokenize function and stored all tokenized words in separate columns

		data	target	target_label
0	\n\nI am sure some bashers of Pens fans are pr...	10	rec.sport.hockey	[I, am, sure, some, bashers, of, Pens, fans, a...
1	My brother is in the market for a high-perform...	3	comp.sys.ibm.pc.hardware	[My, brother, is, in, the, market, for, a, vid...
2	\n\n\n\n\tFinally you said what you dream abou...	17	talk.politics.mideast	[Finally, you, said, what, you, dream, about, ...
3	\nThink!\n\nIt's the SCSI card doing the DMA t...	3	comp.sys.ibm.pc.hardware	[Think, It, the, SCSI, card, doing, the, DMA, ...
4	1) I have an old Jasmine drive which I cann...	4	comp.sys.mac.hardware	[I, have, an, old, Jasmine, drive, which, I, c...

Removing Stop Words

```
1 # Defining function to remove english stopwords from tokenized data with the help of stopwords function of NLTK
2 def remove_stopword(data):
3     stopword = stopwords.words("english")
4     return [word for word in data if not word in stopword]
5
6 # Applying remove_stopword function to column tokenized of dataset and storing returned values into stopword_removed column
7 df['stopword_removed'] = df.apply(lambda x: remove_stopword(x['tokenized']),
8                                     axis=1)
9 df[["data", "stopword_removed"]].head()
```

	data	stopword_removed
0	\n\nI am sure some bashers of Pens fans are pr... [I, sure, bashers, Pens, fans, pretty, confuse...	
1	My brother is in the market for a high-perform... [My, brother, market, video, card, supports, V...	
2	\n\n\n\n\tFinally you said what you dream abou... [Finally, said, dream, Mediterranean, That, ne...	
3	\nThink!\n\nIt's the SCSI card doing the DMA t... [Think, It, SCSI, card, DMA, transfers, NOT, d...	
4	1) I have an old Jasmine drive which I cann... [I, old, Jasmine, drive, I, use, new, system, ...	

- We applied the English stop word removal on the tokenized word using “stopwords.words()” of the NLTK function.

Stemming

- After removing the stop words, we convert the words to their root form, which is called stemming.

```
1 # Converting words to it's root form using stemming technique
2 # Word stemming using PorterStemmer function of NLTK
3 # And return the stemmed word list
4
5
6 def stemming(data):
7     stemmer = PorterStemmer()
8     return [stemmer.stem(word) for word in data]
```

```
1 # Applying stemming function to dataset with column name stopword_removed
2
3 df['stemmed_words'] = df.apply(lambda x: stemming(x['stopword_removed']),
4                                 axis=1)
5 df[['data', 'stemmed_words']].head()
```

	data	stemmed_words
0	\n\nI am sure some bashers of Pens fans are pr...	[i, sure, basher, pen, fan, pretti, confus, la...]
1	My brother is in the market for a high-perform...	[my, brother, market, video, card, support, ve...]
2	\n\n\n\tFinally you said what you dream abou...	[final, said, dream, mediterranean, that, new,...]
3	\nThink!\n\nIt's the SCSI card doing the DMA t...	[think, it, scsi, card, dma, transfer, not, di...]
4	1) I have an old Jasmine drive which I cann...	[i, old, jasmin, drive, i, use, new, system, m...]

Lemmatization

- The only difference between lemmatization and stemming is that stemming removes or stems the last few characters of a word, often leading to incorrect meanings and spelling
- Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
- The final step of the preprocessing is the joining of the lemmatized_words into a string.

```
1 # Function that join and return lemmatized words to string.
2 def word_rejoining(data):
3     return " ".join(data)

1 # Applying word_rejoining function to lemmatized_words and
2 # Storing final string to dataframe with column name final_string
3 df['final_string'] = df.apply(lambda x: word_rejoining(x['lemmatized_words']),
4                               axis=1)
5 df[['data', 'final_string']].head()

          data           final_string
0 \n\nI am sure some bashers of Pens fans are pr... i sure basher pen fan pretti confus lack kind ...
1 My brother is in the market for a high-perform... my brother market video card support vesa loca...
2 \n\n\n\tFinally you said what you dream abou... final said dream mediterranean that new the ar...
3 \nThink!\n\nIt's the SCSI card doing the DMA t... think it scsi card dma transfer not disk the s...
4 1) I have an old Jasmine drive which I cann... i old jasmin drive i use new system my underst...
```

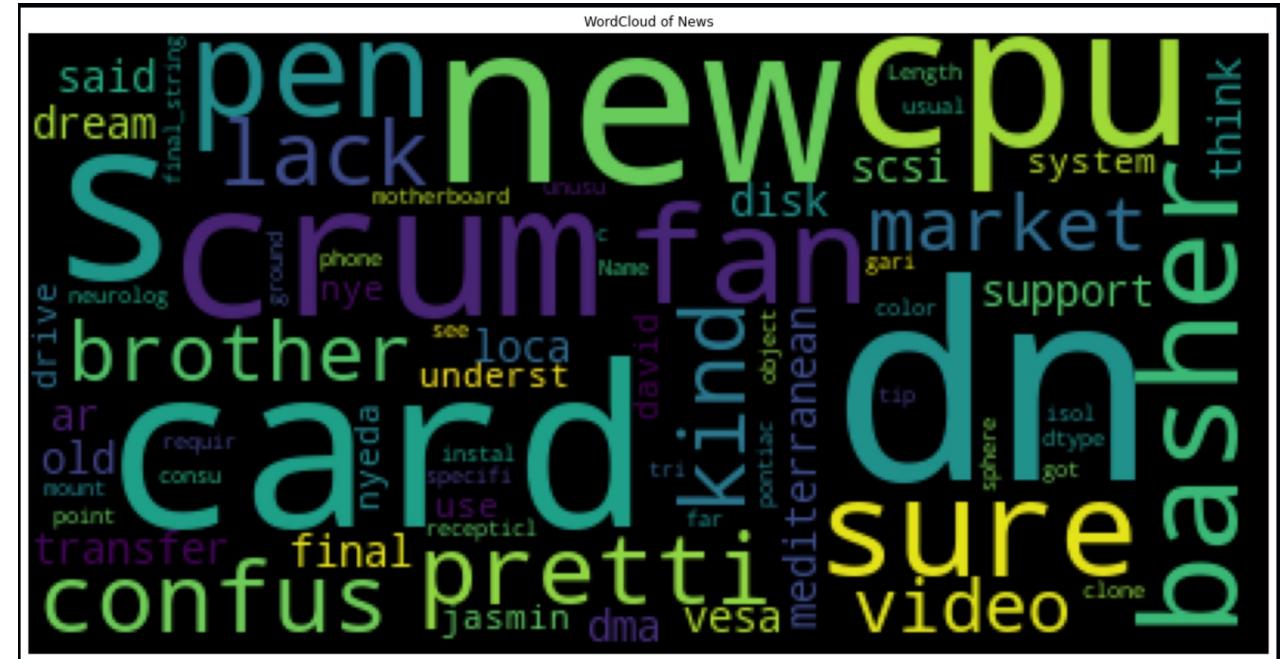
```
1 # lemmatization also convert words to its root form
2 # But the only difference between lemmatization and stemming is that stemming just removes or stems
3 # lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
4
5 def lemmatization(data):
6     lemmatizer = WordNetLemmatizer()
7     return [lemmatizer.lemmatize(word) for word in data]

1 df['lemmatized_words'] = df.apply(lambda x: stemming(x['stopword_removed']),
2                                   axis=1)
3 df[['data', 'lemmatized_words']].head()

          data           lemmatized_words
0 \n\nI am sure some bashers of Pens fans are pr... [i, sure, basher, pen, fan, pretti, confus, la...
1 My brother is in the market for a high-perform... [my, brother, market, video, card, support, ve...
2 \n\n\n\tFinally you said what you dream abou... [final, said, dream, mediterranean, that, new, ...
3 \nThink!\n\nIt's the SCSI card doing the DMA t... [think, it, scsi, card, dma, transfer, not, di...
4 1) I have an old Jasmine drive which I cann... [i, old, jasmin, drive, i, use, new, system, m...
```

WordCloud

- After preprocessing, we again generated a word cloud on `final_string`, and we got a clean and clear view of the words.



Feature Extraction

- Extracting features from the text data is called word vectorization or word embeddings. This is a very crucial part of Natural Language Processing because this feature vector will go into the model training. If this is not done correctly, the model will lead to misprediction.
- Word Vectorization is the technique of converting words to numerical form to perform model training on text-based data.
- For this problem, we have used two-word vectorizing techniques.
- TF-IDF
- Word2vec

TF-IDF

- Term Frequency-Inverse Document Frequency is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.
- This is done by multiplying two metrics: how many times a word appears in a document and the inverse document frequency of the word across a set of documents.
- Here we have used the TfidfVectorizer class of the Sklearn library to convert the words to vectors.
- Here we discovered that from all features, 61.29% of data is non zero, and 39% of features contain zeros because of the uneven length of news text.

```
1 # to store the result of training
2 tf_idf_train_acc = {}
3 # to store the result of test data
4 tf_idf_test_acc = {}

1 # Creating instance of TfIdfVectorizer function of sklearn library
2 tf_idf = TfidfVectorizer()
3 # Converting final_strings to its meaning full words vectors from dataframe
4 # Storing final traing data to X variable
5 X = tf_idf.fit_transform(df["final_string"])
6 # Storing Labels to Y variable
7 Y = df["target"]
8 # Printing Shape of data
9 X.shape
```

(18846, 57689)

```
1 # Displaying average non-zero values from dataset
2 X.nnz / float(X.shape[0])
```

61.29640241961159

```
1 # Printing array of X
2 X.toarray()
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
1 # getting training accuracy of model
2 y_pred = naive_bayes_classifier.predict(X_train)
3 NB_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", NB_train_acc)
5 tf_idf_train_acc["NaiveBayes"] = NB_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = naive_bayes_classifier.predict(X_test)
9 NB_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", NB_test_acc)
11 tf_idf_test_acc["NaiveBayes"] = NB_test_acc
```

Train Accuracy : 0.8114884584770496
Test Accuracy : 0.6896551724137931

```
1 # getting training accuracy of model
2 y_pred = sgd.predict(X_train)
3 SGD_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", SGD_train_acc)
5 tf_idf_train_acc["SGD"] = SGD_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = sgd.predict(X_test)
9 SGD_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", SGD_test_acc)
11 tf_idf_test_acc["SGD"] = SGD_test_acc
```

Train Accuracy : 0.857389227911913
Test Accuracy : 0.7026525198938992

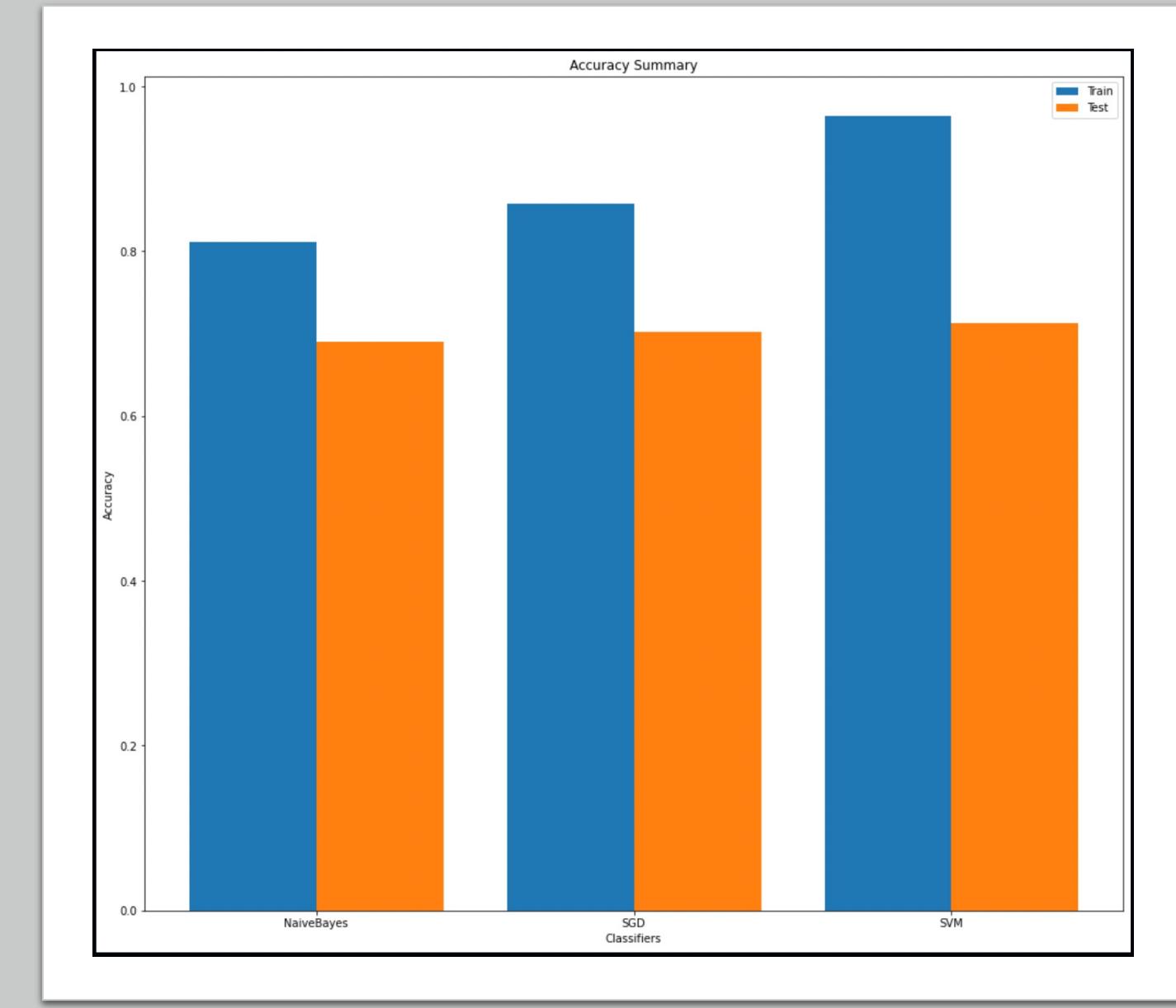
```
1 # getting training accuracy of model
2 y_pred = svm.predict(X_train)
3 SVM_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", SVM_train_acc)
5 tf_idf_train_acc["SVM"] = SVM_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = svm.predict(X_test)
9 SVM_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", SVM_test_acc)
11 tf_idf_test_acc["SVM"] = SVM_test_acc
```

Train Accuracy : 0.964181480498806
Test Accuracy : 0.7124668435013263

- To train and evaluate the model, we split the dataset into train-test with a ratio of 80-20%. h. We have trained the model with three different classifiers.
- Naive Bayes Classifier
- SGD Classifier
- Support Vector Machine(SVM) Classifier

Train-Test Summary

- From all three classification models, Support Vector Machine Classifier performed well with train accuracy of 96% and test accuracy of 71% other than Naïve Bayes with 81%-69% Train-Test and SGD with 85%-70% train-test accuracy.



Word2Vec

- Word2vec is a technique for natural language processing to convert words into vectors. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest different words for a partial sentence.
- Here we have taken the lemmatized_words for training the network.
- We have trained the word2vec model using Gensim's Word2Vec function with vector_size=500 and window=100 with a min_count of 20.
- Generated aggregated sentence vectors based on the word vectors for each word in the sentence
- Compute final sentence vectors by averaging the word vectors for the words contained in the sentence

```
1 # Splitting the lemmatized_words list into train test dataset
2 X_train, X_test, y_train, y_test = train_test_split(df['lemmatized_words'],
3                                                 df['target'],
4                                                 test_size=0.2)
```

```
1 # Train the word2vec model with vector_size 800 and window size 600
2 w2v_model = gensim.models.Word2Vec(X_train,
3                                     vector_size=500,
4                                     window=100,
5                                     min_count=20)
```

```
1 # Generating aggregated sentence vectors based on the word vectors for each word...in_the_sentence
2 words = set(w2v_model.wv.index_to_key)
3 X_train_vect = np.array(
4     [np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_train])
5 X_test_vect = np.array(
6     [np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_test])
```

```
1 # Compute sentence vectors by averaging the word vectors for the words contained in the sentence
2 X_train_vect_avg = []
3 for v in X_train_vect:
4     if v.size:
5         X_train_vect_avg.append(v.mean(axis=0))
6     else:
7         X_train_vect_avg.append(np.zeros(500, dtype=float))
8
9 X_test_vect_avg = []
10 for v in X_test_vect:
11     if v.size:
12         X_test_vect_avg.append(v.mean(axis=0))
13     else:
14         X_test_vect_avg.append(np.zeros(500, dtype=float))
```

Word2Vec Model Training

- We have performed model training with three different classifiers.
 - i. Stochastic Gradient Descent(SGD) Classifier
 - ii. Random Forest Classifier
 - iii. Support Vector Machine(SVM) Classifier

```
1 # getting training accuracy of model
2 y_pred = sgd.predict(X_train_vect_avg)
3 SGD_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", SGD_train_acc)
5 w2v_train_acc["SGD"] = SGD_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = sgd.predict(X_test_vect_avg)
9 SGD_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", SGD_test_acc)
11 w2v_test_acc["SGD"] = SGD_test_acc
```

Train Accuracy : 0.599230565136641
Test Accuracy : 0.5952254641909814

```
1 # getting training accuracy of model
2 y_pred = rf_model.predict(X_train_vect_avg)
3 rf_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", rf_train_acc)
5 w2v_train_acc["RandomForest"] = rf_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = rf_model.predict(X_test_vect_avg)
9 rf_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", rf_test_acc)
11 w2v_test_acc["RandomForest"] = rf_test_acc
```

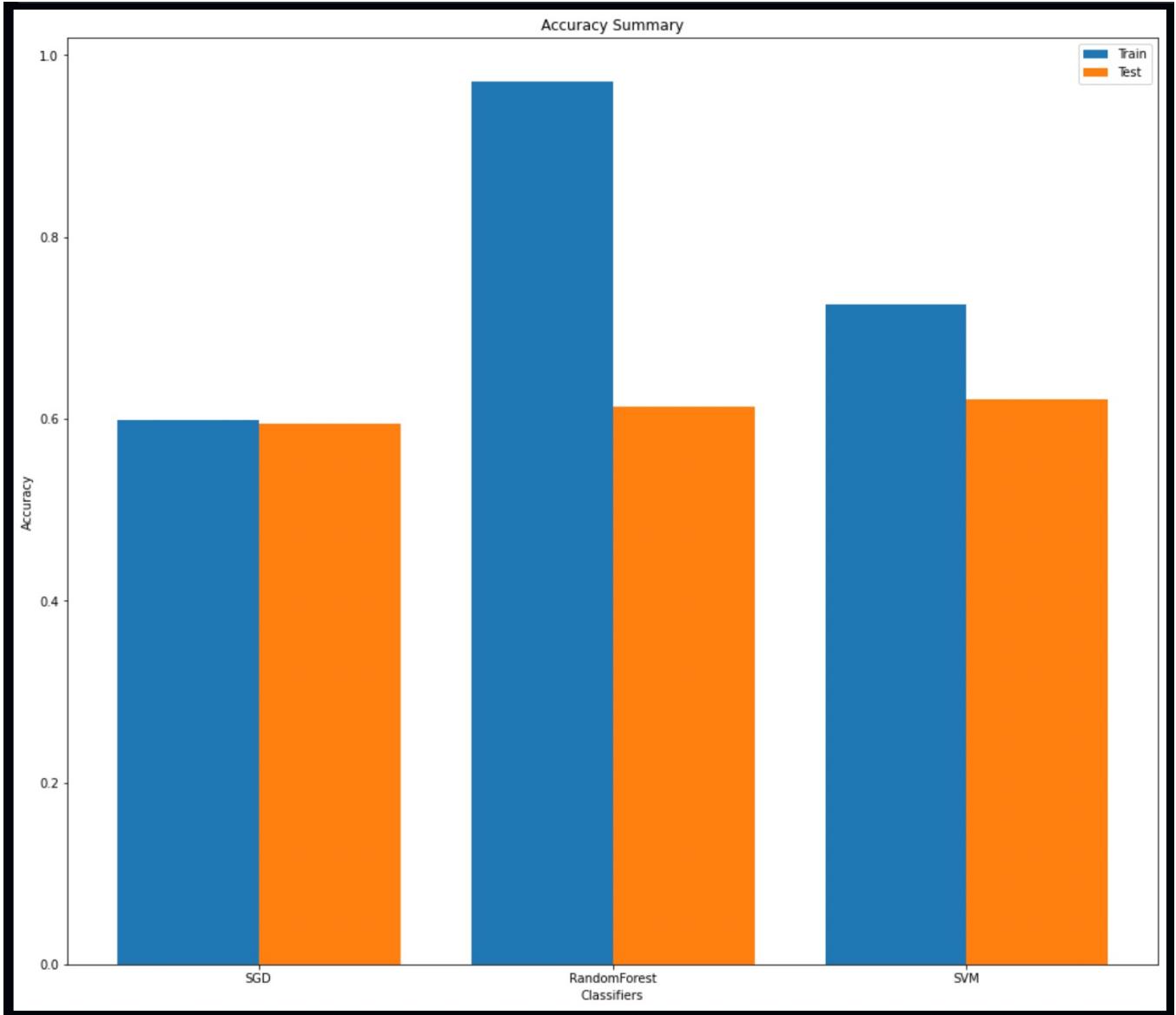
Train Accuracy : 0.9710798620323693
Test Accuracy : 0.6127320954907162

```
1 # getting training accuracy of model
2 y_pred = svm.predict(X_train_vect_avg)
3 SVM_train_acc = metrics.accuracy_score(y_train, y_pred)
4 print("Train Accuracy : ", SVM_train_acc)
5 w2v_train_acc["SVM"] = SVM_train_acc
6
7 # Performing prediction on test dataset
8 y_pred = svm.predict(X_test_vect_avg)
9 SVM_test_acc = metrics.accuracy_score(y_test, y_pred)
10 print("Test Accuracy : ", SVM_test_acc)
11 w2v_test_acc["SVM"] = SVM_test_acc
```

Train Accuracy : 0.7263863093658796
Test Accuracy : 0.6209549071618037

Train-Test Summary

- For the word2vec word embedding technique from all three classification models, the Support Vector Machine classifier performed well with the train-test, 73%-61% accuracy score.



Conclusion

- To conclude, we performed classification on 20 newsgroup datasets with 2 different word embedding techniques with all text pre-processing steps with feature selection to train the classification models.
- We observe that the TF-IDF vectorization technique performed better for this problem, with a 71% accuracy score higher than Word2Vec embedding using the SVM Classifier.
- Because the Word2Vec technique is most suitable for recommendation problems, but this is the problem of Multiclass text classification, so for that TI-IDF is best.



Thank You