Analysis of 20 Newsgroups Dataset Using Various Classification Models and Techniques

Table of Contents

1. Introduction

One of the broadly used natural language processing task in business sector is text classification. Text classification aims to assign predefined classes or categories to a document based on the content, making it easier to manage and analyse. The goal of this project is to classify text documents into defined categories. To achieve that, the ’20 Newsgroups data’ is utilised, a common multi-class NLP dataset. The dataset includes around of 18000 newsgroups documents of 20 topics divided in two subsets: one for training and the other one for testing (performance evaluation). The outline of the project is as follow: necessary pre-processing steps are made before training the machine learning models as well as the challenges faced in this process, an evaluation and discussion based on the experiments and their results and finally a forthcoming work of some potential avenues of investigation which can be consider worthwhile to pursue.

2. Exploratory Data Analysis (EDA)

The exploratory data analysis process was divided into two categories: general pre-processing steps that are common across all vectorizers and models and certain pre-processing steps as options to measure and evaluate model performance with or without them. Accuracy was selected as an evaluation metric of comparison between machine learning models since greater the accuracy, better the model performs on data that has never seen before.

The following general pre-processing steps are applied before any document being an input to train the model:

- Converting to lower case
- Removal of stop words
- Removing any alphanumeric characters
- Removal of punctuations
- Vectorization: TF-IDF vectorizer was used because it gave better results regarding the model accuracy than the Count Vectorizer.

The following steps were added to the above pre-processing process to notice how model performs with these steps:

- Stemming
- Lemmatization
- Using Unigrams/Bigrams

To begin with, the spread of output labels was analysed in the training data to check for any skewness towards the classes. Figure 1 below breakdowns this class distribution. It can be observed that the spread of classes is reasonably even, thus the training dataset is totally balanced. With that being said, the classification models won’t be biased towards any particular category and there is no need for resampling. Also, the figure 2 illustrates the class distribution in the testing data.

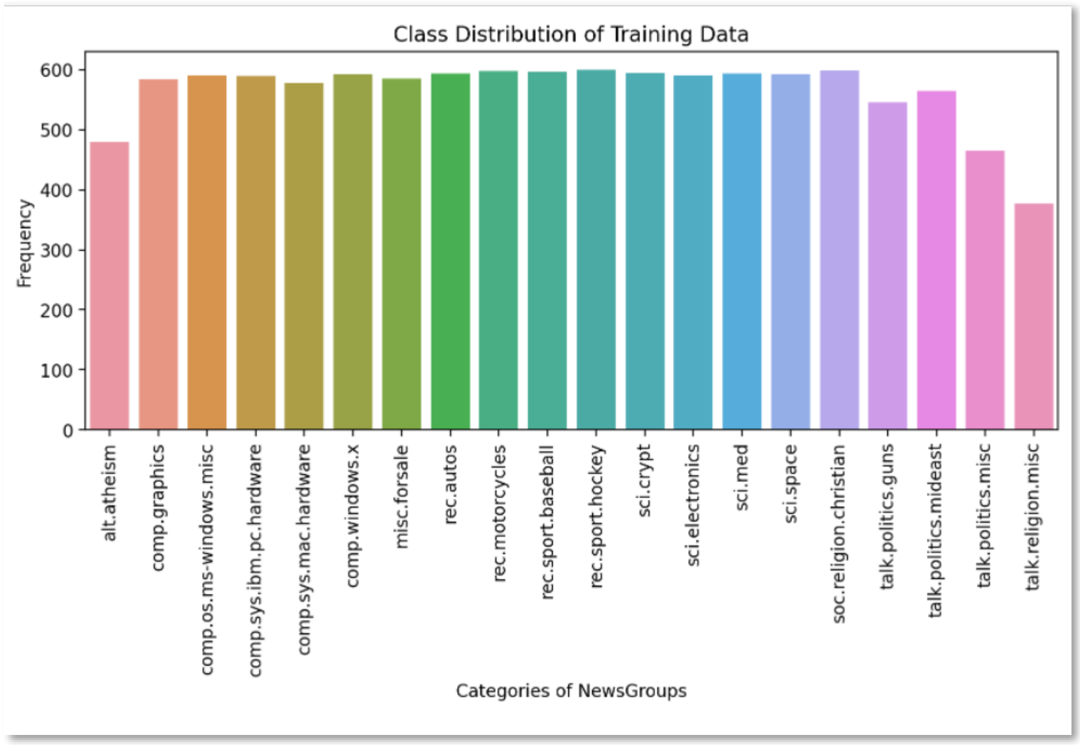


Figure 1: Class Distribution of Training Data.

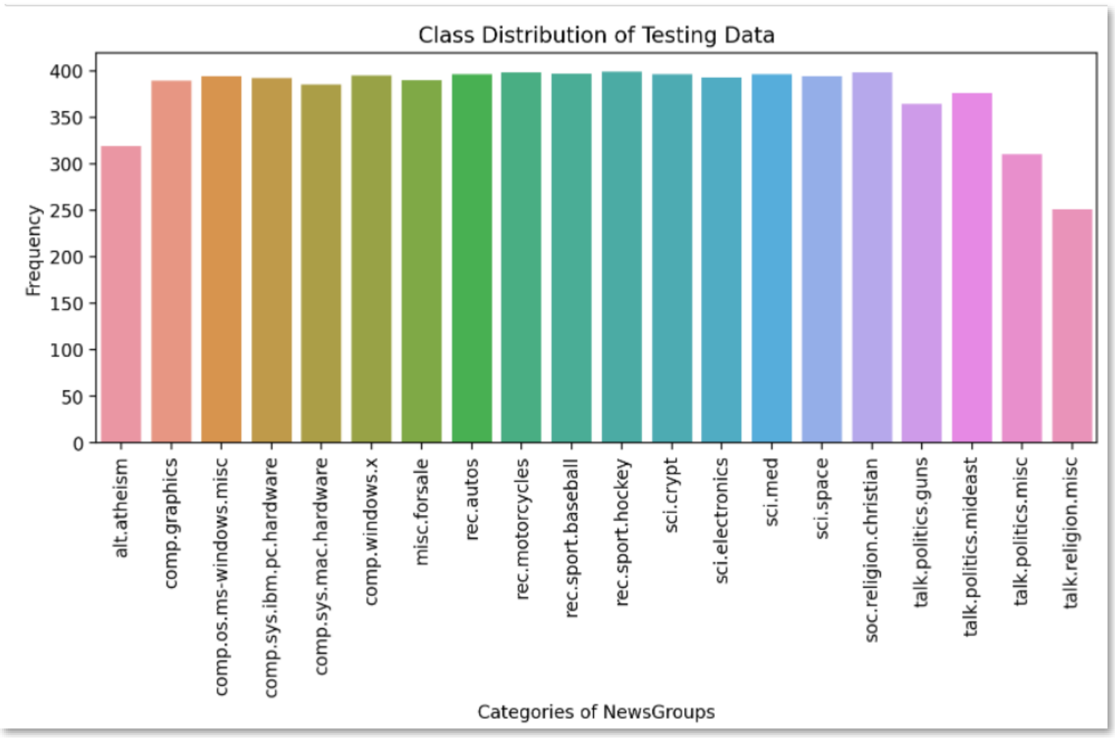
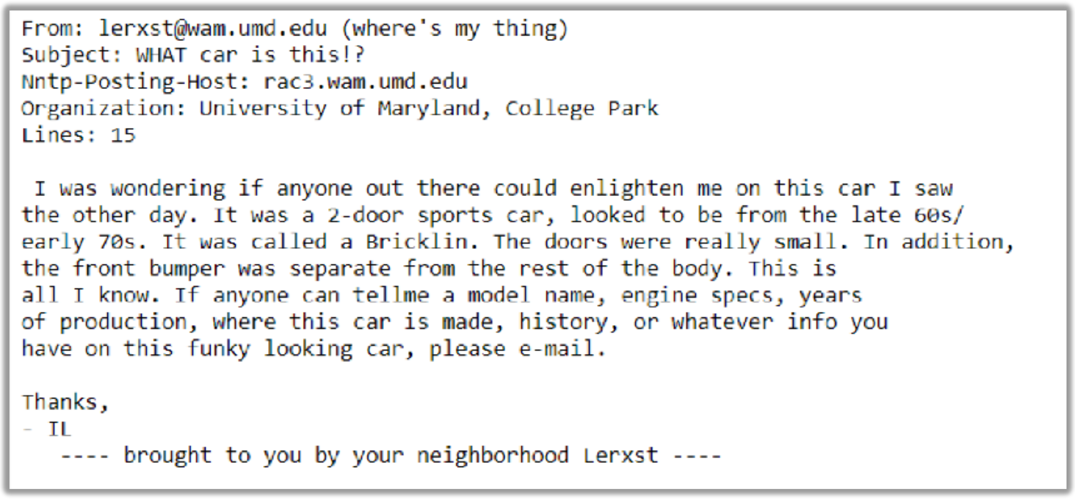


Figure 2: Class Distribution of Testing Data

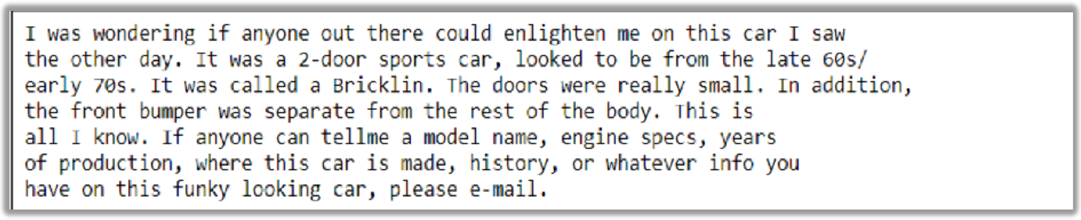
Furthermore, the text data was available with headers, footers and quotes and looked something like:


```
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----
```

However, the functions that load the dataset provide a parameter called remove in a form of tuple which allows to strip out information of each file. The main body of each document was extracted by removing headers, footer, and quotes and eventually the output looked like:


```
I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.
```

Based on scikit learn and 20 Newsgroups dataset documentation, machine learning models have better accuracies (~10% more) with the initial version (maintain headers, footers, and quotes) but this is because classifiers barely have to recognize topics from documents, and they all perform at high level since there is an abundance of clues that differentiate newsgroups. Most writers write on a particular set of topics and the models use that information to classify. This actively demonstrates that classifiers gave less weightage about the content of the text. Therefore, the point is to train a model that takes into account only the words used after the extraction.

3. Methodology

Various features of TF-IDF Vectorizer are compared to find which set works the best under different machine learning models. The modelling procedure starts with the general pre-processing steps in the training and testing data, creating a data frame to store the data by first considering no stemming and lemmatization in the vectorizer. Next, stemming and lemmatization is applied separately in the vectorizer. Within each of these settings, firstly only unigrams are applied as the results from TF-IDF vectorizer and then the results are compared when both unigrams and bigrams are considered in the set. Combining all these steps, four different classification models are trained to evaluate how they perform on the testing dataset. These includes Naïve Bayes, Logistic Regression, Stochastic Gradient Descend Classifier, and k Nearest Neighbors. Sklearn library is used to perform the analysis using Python programming language.

General Pre-processing steps - Data Cleaning

Text cleaning is the most important stage as it can improve the overall performance and efficiency of the models. However, it requires attention to many issues because real world information is very prone to outliers. Documents frequently contain incorrect punctuation, misspelled words, and other mistakes (Khaled Albishre, Mubarak Albathan, Yuefeng Li, 2015). Therefore, in this phase, punctuation, capital letters, numbers and stop words are removed from the corpus to perform data cleaning. Punctuation removal is an essential NLP pre-processing step because it can be remarkably dangerous and affects the outcome of any text processing procedure. So, it is necessary to get rid of these parts of the data or noise by removing punctuation marks. Regex expression is used to make this happen.

Moreover, stop words can be referred as the most widely recognised words in a language and such words like ‘is’, ‘which’, ‘the’ etc. reduce the effectiveness of text data documents. Therefore, stop words removal play a crucial role for text pre-processing steps for the main reason that it decreases the noise of the document, making the process more effective and efficient. To sum up, the goal of data text cleaning is to reduce the dimensionality to manage the number of terms in the document (Yuefeng Li, Abdulmohsen Algarni, Mubarak Albathan, Yan Shen, and Moch Arif Bijaksana, 2015).

Not performing Stemming and Lemmatization in the TF-IDF Vectorizer

First, a count vectorizer is created to convert text document collection to numerical data (matrix of integers). Count Vectorizer helps to generate a sparse matrix of word counts known as bag of words which simply is a text representation that shows the occurrence of words within a document and the vocabulary of them as well (Santanu Kumar Rath, Ankit Agrawal, Abinash Tripathy, 2016). Then, TF-IDF Vectorizer is created which stands as term frequency – inverse document frequency. Term frequency is the raw count of a particular term in the text while inverse document frequency is the occurrence of any word in all documents. This method allows to understand the content of words across an entire corpus of documents instead of just its relative importance in a single document. Using a Multinomial Naïve model, these two methods are compared each other to select which vectorizer performs better. TF-IDF Vectorizer had better performance and by trials, applying sublinear tf scaling $(1 + \log(tf))$ improves overall results.

The first experiment was to form the vectorizer extracting only unigrams from the text data and test the four machine learning algorithms. For logistic regression and k Nearest Neighbors a grid search was used to tune their hyperparameters. Also, an elbow method was created for choosing a reasonable k value apart from the grid search.

For Logistic Regression, the parameter ‘penalty’ was tuned with values {‘L1’,’L2’} norm. L2 norm gave the best results in the penalization. For Knn, the parameter ‘n_neighbors’ was tuned with values {5,10,100,200} and ‘weights’ parameter with values {‘uniform’, ‘distance’}. Best performance for kNN is achieved with 5 ‘n_neighbors’ and ‘weights’ as ‘distance’.

Next, unigrams as well as bigrams were extracted from the text data to form the vectorizer. N-gram model represent a sequence of N words as a singular item in the vocabulary. This method helps to predict the next item in sentence, meaning that provides a way to quantify word-context (Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat, Robert Nowak, 2008). Then, TF-IDF vectorizer is generated from these n-gram representations. Similarly, the four classification models are tested, and the results can be observed in table 1.

Table 1: Performance of classifiers on test data.

ngrams	Classifier Accuracy (%)			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**kNN**
Unigram	69.7	69.0	69.5	8.57
Unigram+Bigram	69.1	68.3	70.3	7.9

Applying only Stemming in the TF-IDF Vectorizer

Stemming technique is an important method in data cleaning which reduces inflectional or derived words to their word root/stem (Sandesh Gharatkar Aakash Ingle Tanmay Naik Ashwini Save, 2017). Snowball English Stemmer algorithm from the NLTK package in python is used and a class is defined to represent the stemmer. Snowball stemmer is also known as porter2 stemmer. It is a better version of it since porter stemmer had some issues that re fixed (Divya Khyani, Siddhartha ,Niveditha,Divya, 2020).

Again, classification models are tested on data that have never seen before and table 2 breakdowns the performance of each model extracting unigrams and then both unigrams and bigrams to form the vectorizer.

Table 2: Performance of classifiers using stemming technique.

ngrams	Classifier Accuracy (%)			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**kNN**
Unigram	69.5	69.0	69.8	8.4
Unigram+Bigram	68.9	68.3	70.7	8.3

Applying only Lemmatization in the TF-IDF Vectorizer

Finally, lemmatization is applied. Lemmatization is another data cleaning method which aims to take into account various inflected forms to be analysed as a singular unit, related them back to their shared base lemma. In other words, with this method, we are getting grammatically correct normal form of a word with the use of morphology (Vimala Balakrishnan, 2014). Word Net Lemmatizer is used from NLTK package and POS (part-of-speech) word tagging and a class is defined to represent Lemma Tokenizer. The performance of machine learning is evaluated, first with unigrams and then with both unigrams and bigrams in the vectorizer. Results are shown in table 3.

Table 3: Performance of classifiers on test data using lemmatization.

ngrams	Classifier Accuracy (%)			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**kNN**
Unigram	69.5	69.1	70.0	8.53
Unigram+Bigram	68.8	68.4	70.4	8.1

4. Results and Discussions

In this section, we compare the results that we achieved in the previous section. Table 4 summarizes the best performance from each classifier. It can be observed that Multinomial Naïve Bayes Model and k Nearest Neighbors have their best performance when neither stemming nor lemmatization and only unigrams are extracted in the TF-IDF vectorizer. Logistic regression performs the best when lemmatization is used and only unigrams are extracted in the TF-IDF vectorizer. Finally, it can be seen that Stochastic Gradient Descend Classifier has the best performance when stemming is applied with both unigrams and bigrams are extracted in the TF-IDF vectorizer.

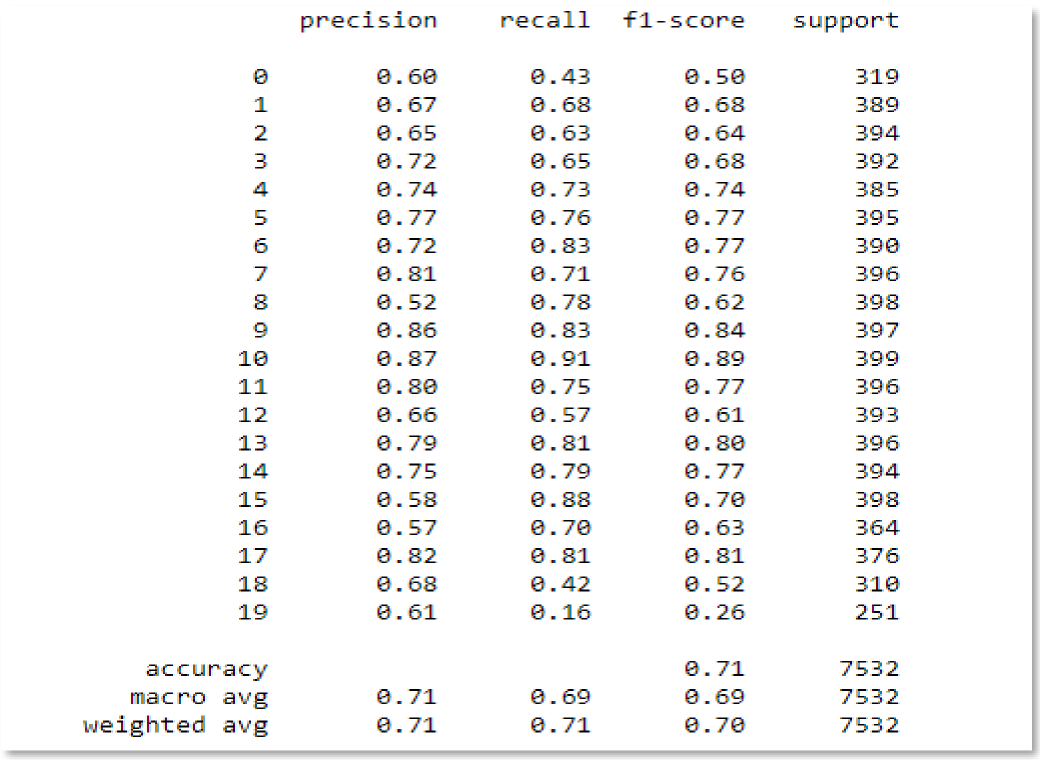
The best overall performance is achieved from Stochastic Gradient Descend Classifier with 70.7% accuracy, followed by Multinomial Naïve Bayes and Logistic Regression model. K Nearest Neighbors model has extremely poor performance and this is because this machine learning model suffers from the curse of dimensionality when number of features are large (Rojas, 2015). The curse of dimensionality refers to the problem of the increasing data dimensions resulting in high computational efforts to process or analyse the data. The more features, the more data points are needed to fill space. Therefore, knn's distance measure becomes meaningless when the dimension of the data increases significantly.

Table 4: Summarizing the best performance from each classifier

Classifier	Model	Test Accuracy (%)
Multinomial Naïve Bayes	No stemming and lemmatization +Unigrams	69.7
Logistic Regression	With lemmatization and Unigrams	69.1
SGD Classifier	With stemming + Unigrams and Bigrams	70.7
K Nearest Neighbors	With lemmatization and unigrams	8.57

Moreover, figure 3 illustrates the evaluation metrics for each of the 20 categories for the best SGD classifier. To be more specific, precision, recall and f1 score are shown in this figure. Precision is a measure of how many of the positive predictions made are correct while recall is a measure of how many of the positive cases the SGD classifier correctly predicted, over all the positive cases in the data. Lastly, f1 score is the harmonic mean of precision and recall.

Finally, a confusion matrix of the best SGD classifier is displayed in figure 4. For classification, a confusion matrix is a table that describes the performance of the model, meaning that how good is the model at determining the positive and negative class/outcome.

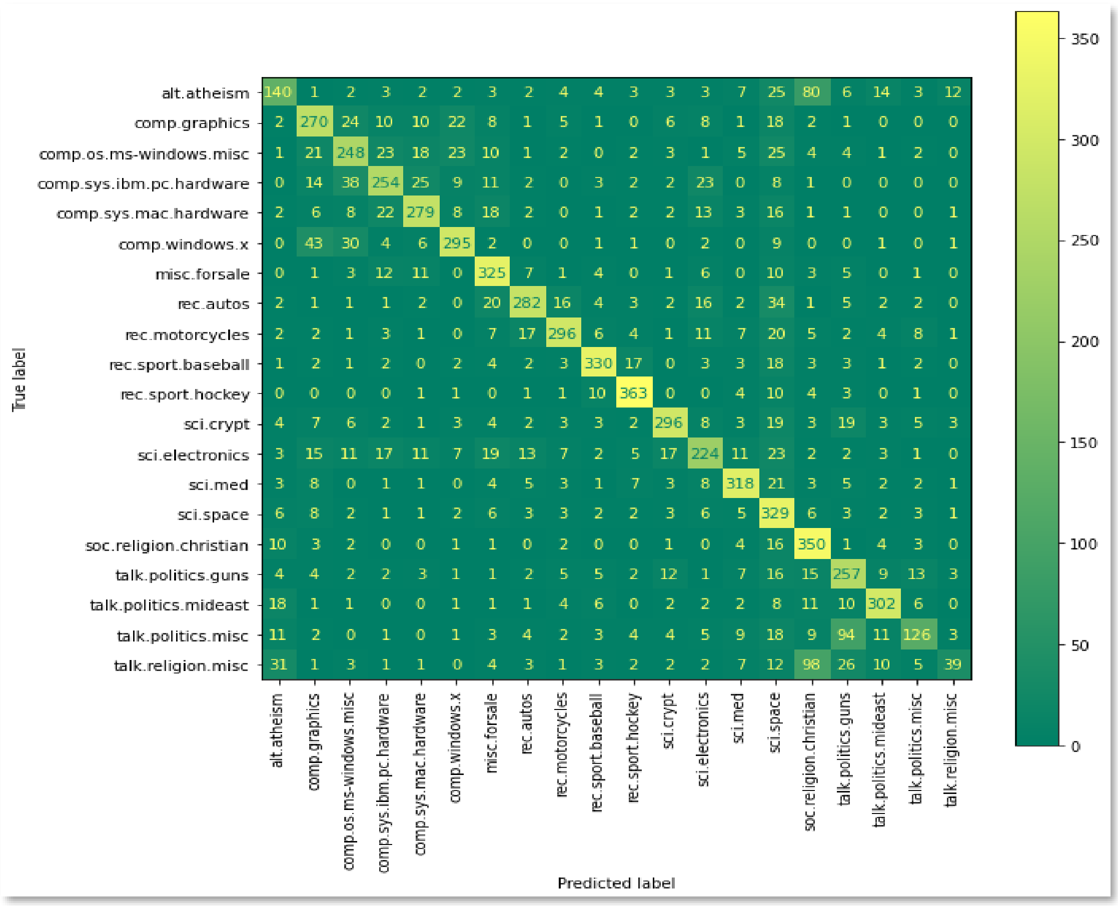


```
              precision    recall  f1-score   support

           0       0.60      0.43      0.50       319
           1       0.67      0.68      0.68       389
           2       0.65      0.63      0.64       394
           3       0.72      0.65      0.68       392
           4       0.74      0.73      0.74       385
           5       0.77      0.76      0.77       395
           6       0.72      0.83      0.77       390
           7       0.81      0.71      0.76       396
           8       0.52      0.78      0.62       398
           9       0.86      0.83      0.84       397
          10       0.87      0.91      0.89       399
          11       0.80      0.75      0.77       396
          12       0.66      0.57      0.61       393
          13       0.79      0.81      0.80       396
          14       0.75      0.79      0.77       394
          15       0.58      0.88      0.70       398
          16       0.57      0.70      0.63       364
          17       0.82      0.81      0.81       376
          18       0.68      0.42      0.52       310
          19       0.61      0.16      0.26       251

    accuracy                           0.71      7532
   macro avg       0.71      0.69      0.69      7532
weighted avg       0.71      0.71      0.70      7532
```

Figure 3: Evaluation metrics for the best SGD Classifier.



True label \ Predicted label	alt.atheism	comp.graphics	comp.os.ms-windows.misc	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	comp.windows.x	misc.forsale	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey	sci.crypt	sci.electronics	sci.med	sci.space	soc.religion.christian	talk.politics.guns	talk.politics.mideast	talk.politics.misc	talk.religion.misc
alt.atheism	140	1	2	3	2	2	3	2	4	4	3	3	3	7	25	80	6	14	3	12
comp.graphics	2	270	24	10	10	22	8	1	5	1	0	6	8	1	18	2	1	0	0	0
comp.os.ms-windows.misc	1	21	248	23	18	23	10	1	2	0	2	3	1	5	25	4	4	1	2	0
comp.sys.ibm.pc.hardware	0	14	38	254	25	9	11	2	0	3	2	2	23	0	8	1	0	0	0	0
comp.sys.mac.hardware	2	6	8	22	279	8	18	2	0	1	2	2	13	3	16	1	1	0	0	1
comp.windows.x	0	43	30	4	6	295	2	0	0	1	1	0	2	0	9	0	0	1	0	1
misc.forsale	0	1	3	12	11	0	325	7	1	4	0	1	6	0	10	3	5	0	1	0
rec.autos	2	1	1	1	2	0	20	282	16	4	3	2	16	2	34	1	5	2	2	0
rec.motorcycles	2	2	1	3	1	0	7	17	296	6	4	1	11	7	20	5	2	4	8	1
rec.sport.baseball	1	2	1	2	0	2	4	2	3	330	17	0	3	3	18	3	3	1	2	0
rec.sport.hockey	0	0	0	0	1	1	0	1	1	10	363	0	0	4	10	4	3	0	1	0
sci.crypt	4	7	6	2	1	3	4	2	3	3	2	296	8	3	19	3	19	3	5	3
sci.electronics	3	15	11	17	11	7	19	13	7	2	5	17	224	11	23	2	2	3	1	0
sci.med	3	8	0	1	1	0	4	5	3	1	7	3	8	318	21	3	5	2	2	1
sci.space	6	8	2	1	1	2	6	3	3	2	2	3	6	5	329	6	3	2	3	1
soc.religion.christian	10	3	2	0	0	1	1	0	2	0	0	1	0	4	16	350	1	4	3	0
talk.politics.guns	4	4	2	2	3	1	1	2	5	5	2	12	1	7	16	15	257	9	13	3
talk.politics.mideast	18	1	1	0	0	1	1	1	4	6	0	2	2	2	8	11	10	302	6	0
talk.politics.misc	11	2	0	1	0	1	3	4	2	3	4	4	5	9	18	9	94	11	126	3
talk.religion.misc	31	1	3	1	1	0	4	3	1	3	2	2	2	7	12	98	26	10	5	39

Figure 4: Confusion Matrix of the best SGD Classifier.

5. Future Work

Based on the outcome of the experiments, there are some more potential avenues of investigation. These methods are more hybrid and robust than most of the most machine learning classifiers. Neural Networks can also be studied to check how they perform on this data. The same pre-processing steps can be made to finally form the TF-IDF vectorizer, but this time the 'min_df' can be restricted to 0.0005. The main reason to do this was to speed the training phase of neural networks because it would take so much time with so many features. Another thing to notice when training neural networks is the problem of overfitting. This can be happened when too many hidden layers are added.

However, these algorithms cannot face the problem of data sparsity due to the lack of better data representation techniques. This is where deep learning plays crucial role. Deep learning-based text classification solves the problem of data sparsity by word embeddings which capture syntactic as well as semantic information of the text data. This kind of methodology is extremely successful capturing the contextual information of the data and resolve the problems of data sparsity and thus, they outperform state-of-the-art machine learning based classification approaches (Muhammad Nabeel Asim, Muhammad Usman Ghani Khan, Muhammad Imran Malik, Andreas Dengel,Sheraz Ahmed, 2019).

6. Conclusion

To conclude, four classification models are tested on data that have never seen before in order to fairly evaluate their performance. After some general pre-processing steps, count vectorizer and TF-IDF vectorizer were used to check which of these two perform the best. TF-IDF vectorizer gave better results and the numerical data was used as an input to 'feed' the supervised machine learning models. Also, lemmatization helped the SGD classifier achieve the best performance with 70.7% overall accuracy. Also, unigrams helped to achieve higher accuracy using Multinomial Naïve Bayes, Logistic Regression, and k Nearest Neighbors model. Finally, some future work was introduced to the report if the project were to be continued.

References

Divya Khyani, Siddhartha ,Niveditha,Divya, 2020. An Interpretation of Lemmatization and Stemming in Natural Language Processing. *Journal of University of Shanghai for Science and Technology .*

Khaled Albishre, Mubarak Albathan, Yuefeng Li, 2015. Effective 20 Newsgroups Dataset Cleaning. *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.*

Muhammad Nabeel Asim, Muhammad Usman Ghani Khan, Muhammad Imran Malik, Andreas Dengel,Sheraz Ahmed, 2019. A Robust Hybrid Approach for Textual Document Classification. *2019 International Conference on Document Analysis and Recognition (ICDAR).*

Rojas, R., 2015. The Curse of Dimensionality.

Sandesh Gharatkar Aakash Ingle Tanmay Naik Ashwini Save, 2017. Review Preprocessing Using Data Cleaning And Stemming Technique. *International Conference on Innovations in information Embedded and Communication Systems (ICIIECS).*

Santanu Kumar Rath, Ankit Agrawal, Abinash Tripathy, 2016. Classification of sentiment reviews using n-gram machine learning approach. *Department of Computer Science and Engineering, National Institute of Technology Rourkela, India.*

Vimala Balakrishnan, E. L.-Y., 2014. Stemming and Lemmatization: A Comparison of Retrieval Performances. *Lecture Notes on Software Engineering,* 2(3).

Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat, Robert Nowak, 2008. Learning Bigrams from Unigrams.

Yuefeng Li, Abdulmohsen Algarni, Mubarak Albathan, Yan Shen, and Moch Arif Bijaksana, 2015. Relevance Feature Discovery for Text Mining. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* 27(6).