Natural Language Processing: 20 Newsgroups Document Classification

Rohit Sunku
210008043
Data Science MSc
rohit.sunku@city.ac.uk

1 Problem Statement and Motivation

The primary goal of this project is to conduct a Document Classification on this 20 Newsgroup dataset, partitioned across 20 different news groups. Each document is linked to a specific newsgroup, and the goal is to build a model which will analyse the text of the article, and based on that content, correctly identify which Newsgroup this article is based on. The key motivation behind the 20 Newsgroups dataset was to be used as a popular data set for experiments in applications of Machine Learning techniques, such as text classification and text clustering.

We are attempting to produce a comparison of Large Language Models, traditional Machine Learning models and Deep Learning Models in this context. There has been a lot of research into LLMs, and Machine Learning algorithms, however the prospect of comparing neural networks in this domain, which have been used in NLP papers and experiments before will be the key motivation. The key goal is to suggest a baseline model based on this paper which will be the strongest performing model, that can be utilised by Newspaper agencies, and optimised for their use cases.

When referring to the business context, we can address the beneficiaries of Document Classification will be Linguistic Researchers, News Agents and any organisation that wants to organise textual data. An application can be built using this model, to place real-time textual data into different locations or files, based on their content, allowing for a greater streamlined approach in allocating textual data. For example, Company XYZ's business model features News Articles based on category. News is coming in daily, from different sources (The Web, various articles, podcasts, journals and television). Data can be sourced and confined into mini-documents, which can then be classified based on their category. Journalists at XYZ can then search for their assigned category (e.g. Sport) and then use generative AI and their own expertise to produce a detailed report based on a series of resources as a final product.

Figure 1: sci-med NewsGroup WordCloud

Figure 2: sci-space Newsgroup WordCloud

We gauge the most common words per each NewsGroup, that can be displayed in the form of Word Clouds. Common words include "edu" and "article", which are key words that models use to classify the articles.

2 Research Hypothesis

The aim of our investigation is:

What is the most accurate model for clas-

sifying documents into different newsgroups?

This can be broken down and split into multiple research questions:

- How do Deep Learning algorithms fare in Document Classification?
- The Naive Bayes model is usually the preferred algorithm for textual data. Does it outperform all the other ML algorithms?

These hypotheses are pivotal in summarising the underlying research of this paper. While there are many different approaches which can be taken, with a lot of different analyses being done already, an investigation into a comparison of a series of different models is yet to be seen. These hypotheses aim to address gaps in research from our preliminary literature review outlined in Section 3. It would be useful to provide a state-in-the-art model which can be used for use cases in companies which are similar to this case, which can then be optimised on their terms.

Based on our hypotheses, we predict the following:

1. Large Language Models (LLMs) will outperform Machine Learning algorithms, achieving a high accuracy, due to the complex nature of the Transformer Architecture, analysing stronger dependencies between words and deciphering patterns that the simple framework and methodology of ML algorithms cannot.
2. Deep Learning Algorithms will fare well in Document Classification, however they may struggle with the limited amount of data for all the different classes.
3. The Naive Bayes Algorithm is a state-of-art Machine Learning algorithm used for textual data, and it will outperform all other Machine Learning Models.

3 Related Work and Background

There is a plethora of research papers relating to this popular dataset. We explore papers written in the "paperswithcode", "ieeexplore" and "research gate".

Prior work encompassed a series of articles from well-known sites such as "medium" and "towardsdatascience", with Authors alluding their methods of working through this problem, with some articles providing a pathway for us to ingest the data into our dataset, and conduct a simple analysis. These articles proved useful when starting the task. One key article on medium, written by Rox. S involved extracting features from the text dataset, using the "Bag of Words" method, which involved splitting each file into single words, then counting the number of words appearing within each file. (**?**)

Kim Dongwha. et al (2019) proposes a framework that identifies features using TF-IDF Vectorisation, Latent Dirichlet Allocation and the Doc2Vec model. These 3 feature-extraction frameworks are adopted and compared on training Naive Bayes and the Random Forest classifier. Results show that TF-IDF selected the most significant terms for classification tasks, however it has a more sparse representation than the other two document representation methods. (Kim et al., 2019). Training is completed on 5 different datasets, with the length of the documents in the corpus varying from short sentences to longer documents. (Kim et al., 2019)

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a recently introduced language representing model based upon the transfer learning paradigm (Pappagari et al., 2019). Results showed to be high for BERT architectures, achieving over 90% suggesting that it would be a worthwhile investment within this investigation. (Devi, 2020) investigates the Bag of Words Approach (BoW) and the N-Gram Approach on the Support Vector Machine, Naive Bayes, K-Nearest-Neighbours (KNN) and the Convolutional Neural Network (CNN) model. Specific focus was on comparing training times, with Neural Networks requiring more computational resources in order to run. Another finding in this paper was that the Bag Of Words Vectorisation approach was much quicker and slightly more accurate in comparison with the N-Grams approach. The paper concludes that the SVM model + BoW approach was the best for this dataset. Another notable paper which explores the 20 Newsgroup Dataset specifically, written by (Asim, 2024n) investigates different variations of CNN for Document Classification. Asim et al. (2024) investigates a two stage classification

methodology for text classification, employing a filter-based feature selection algorithm initially (NDM) to develop noiseless vocabulary. The first modification of the CNN is a standard CNN model, with the entire vocabulary of each dataset obtained after pre-processing, however the other version ranks the vocabulary of each class using the filter-based feature selection algorithm (NDM), with only top k ranked features of each class being selected to feed the embedding layer of the model. Results showed that the TSCNN version achieved a 0.917 accuracy, which outperformed every text classification system on this dataset before it.

Lastly, the final paper we investigate focuses on comparing text classifiers on news articles. Pradhan et al. 2017 conducted a similar study to ours, investigating ML algorithms such as Naive Bayes and SVMs. However, one unique algorithm thrown into the mix was the Rocchio Classifier, which is based on a "relevance-feedback" method, where documents are assigned based on relevance. Data is assigned to the class whose centroid is nearest to the data, following a slightly similar approach to the KNN algorithm. Findings proved that the SVM classified news articles into their newsgroups with the highest accuracy in comparison to all other models within the investigation. Naive Bayes followed, coming in second place with a smaller gap on smaller datasets, with lower classes. As the number of classes increase, the accuracy gap between the SVM and other ML algorithms widened.(Pradhan et al., 2022)

Our approach is slightly different, since we investigate how Count Vectorizers perform as a feature-selector, and we compare models from ML, DL and the Transformer family. We also include a larger number of models, however we do not include many variations of them, and only specifically train these on one dataset. Our approach is the most similar Pradhan et al. 2017 (Pradhan et al., 2022), as this paper also features a high number of models being trained.

3.1 Accomplishments

Here is the link to my work: Click here to open this notebook. In terms of the work aims that we have set out prior to this task, we can list these below as follows:

1. Data Loading/ Pre-Processing - Converting the string to lowercase, removing any whitespaces, one letter characters, stop-words (*NLTK*), punctuation (except commas and full-stops) and all numbers.

2. Tokenisation/ Lemmatisation - Use the *NLTK* library to tokenise words and sentences in separate columns. Proceed to lemmatise all words within subject and content columns to the root word.

3. EDA (Exploratory Data Analysis) - Word Clouds, Data Imbalance, TF-IDF Most Frequent/ Less Frequent Words, average number of words and sentences per newsgroup, most common uni-grams, bi-grams and tri-grams per NewsGroup

4. COMPLETED: Machine Learning - Experiment with different algorithms but the same pipeline structure (Count Vectorizer, TF-IDF Transformer)

    - DummyClassifier (BASELINE MODEL 1)
    - Decision Tree (BASELINE MODEL 2)
    - Random Forest
    - Linear SVC
    - Light GBM
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Naive Bayes
    - Gradient Boosting
    - XG Boost
    - K-Nearest Neighbours
    - Perceptron
    - Bagging Classifier

5. COMPLETED: Machine Learning - Experiment with different Pipelines

    - Adding a Normaliser to the Pipeline (Count Vectoriser + TF-IDF Transformer + Normaliser + ML Algorithm)
    - Changing the Vectoriser of the Pipeline (Hashing Vectoriser + Normaliser + ML Algorithm)
    - New Pipeline (Hashing Vectoriser + TF-IDF Transformer + Normaliser + ML Algorithm)
    - Customising parameters on Vectoriser and Transformer

6. COMPLETED: Calculate all the metrics, confusion matrices, classification metrics for the top 4 ML models
7. COMPLETED: Deep Learning - Train and experiment with different models using *TensorFlow* library.
    - Creating a Dummy 1D CNN
    - CNN 1D (Convolutional Neural Network Training)
    - Bi-Directional LSTM (Train on this)
8. FAILED: Transformer Architecture - Train DistilBERT on this classfication problem
9. COMPLETED: Final Comparision of all Models and their outputs
    - How do our trained models compare to our baseline models?
    - Do our results support our hypotheses?

4 Approach and Methodology

Our approach follows a standard Natural Language Processing pipeline, which follows by downloading the data, pre-processing the content and subject headers, appending this together into one column to make the X variable, then the y variable will be the target variable, which contains all 20 newsgroups. We prepare our data for models, by splitting the data into training and testing sets at a split of 70:30. For some models, such as our Deep Learning Algorithms and XGBoost Classifier, these do not take in classes as a string format, so we apply to use a Label Encoder to encode our target variable (Y) into a numerical format.

The core idea behind our approach is to identify which key words/ phrases are in common and correspond to the Newsgroup. We aim to find the intricacies that differentiate these Newsgroups, as these nuances can lead classifying textual data based on its content within a class. This is a Text/ Document Classification task, and our approach is efficient in handling this. Including the subject column gives light to patterns of specific phrases based on their Newsgroup which can be picked up by the model for improved classification.

Our approach is to provide a model comparison on Text Classification, and demonstrate whether incorporation of Neural Networks is necessary than solving this business problem. Key Deep Learning Models that we investigate are:

4.0.1 Bi-Directional LSTM

We make use of Bi-Directional LSTMs which are a variation of the LSTM (Long-Short Term Memory) linked to a type of Recurrent Neural Network (RNN) architecture which extends the capabilities of traditional LSTMs. LSTMs were initially designed for the purpose of overcoming the vanishing gradient problem. This is a key issue in Recurrent Neural Network Architecture, where the gradients do not get updated during back-propagation of sequential data. The size of the gradient will decrease over time, and the weight parameters are not updated anymore due to the insignificant change within the gradients. LSTMs leverage gating mechanisms to control information flow and gradients, preventing them from vanishing as the network is allowed to learn and retain information over longer sequences. (Baeldung, 2024)

The input sequence which is fed into both forward and backward LSTM layers, with the forward LSTM layer processing the sequence in normal chronological order, and the backward LSTM layer processing the sequence in reverse order. Both these layers are concatenated for each time step and concatenated output can be fed into dense layer or another LSTM layer depending on the task.(IBM, 2023)

There are a few limitations of our approach. Firstly, we do not investigate all possible combinations of algorithms and all models, so it will be difficult to determine the most accurate model for this dataset. Furthermore, within our pre-processing, despite us removing stop-words, we can see common words like many, article and one, which are all words that do not have any significant meaning. ML Algorithms and Feature Extractors will still have some noise within the datasets, when identifying the key words and phrases which differentiate newsgroups.

4.0.2 Baselines, Machine Learning Pipelines and Deep Learning Models

The baseline models we train for Machine Learning are the Dummy Classifier and the Decision Tree. Our first baseline model fails, however the Decision Tree does pass the benchmark of 50% accuracy, but only by a small margin. We investigate

the Decision Tree, Random Forest, Linear SVC, Light GBM, Logistic Regression, Support Vector Machine (SVM), Naive Bayes, Gradient Boosting, XGBoost, AdaBoost, K-Nearest-Neighbours (KNN), Perceptron and Bagging Classifier within our Machine Learning Section. You can see the implementations within this notebook here:

We investigate different feature selectors, including the Hashing Vectorizer, Count Vectorizer, TF-IDF Transformer, TF-IDF Vectorizer and a Normaliser. The Hashing Vectorizer uses a hash function to convert textual data into a fixed-size vector, however this induces noise collisions. We found the TF-IDF Transformer combined with the Count Vectorizer was the best approach, and we used this pipeline to compare all of our ML models. After selecting our top 4 models, we compare different hyper-parameter combinations on our best performing Machine Learning model, the Linear SVC, to see how they would effect accuracy.

4.0.3 Working Implementations

We managed to complete two working implementations for each section, Machine Learning and Deep Learning. The general libraries we use within this project were *nltk* which was primarily for textual data pre-processing, *pandas* for data loading, feature engineering and general data pre-processing and*matplotlib* for plotting graphs of our results. The libraries used to build our Light SVM model were *sci-kit learn* used for Machine Learning and *tensorflow* for Deep Learning. The two final models we have are the baseline CNN 1D model, and the Linear SVC.

5 Dataset

The key benefit in using datasets within data science is their ability to providing a real-world context for analysis. Despite being simulated, they offer a better representation of real scenarios faced by businesses (van der Duim, 2023). Analysing a dataset initially, can lead to building a pre-processing pipeline to improve the data quality, increasing the accuracy of our models. Discrepancies such as null values, duplicate values, incorrect data-types and class imbalance (for classification problems) need to be addressed with techniques such as null value implementation, and synthetic sampling. Other methods include normalisation of numerical data into a standardised format reducing the impact of outliers and improving the accuracy and stability of statistical models (Bhandari, 2024).

After a general analysis on our data downloaded and parsed within our .gzip file, we showcase the report here: We also observe the

Total Number of Observations	19997
Total Number of Files	2
Total Number of Variables	20
Total Number of Numeric Features	2
Total Number of Categorical Features	18
Total Number of Null Values	0
Total Number of Duplicate Values	532
Base Format of the Data	.gzip file

Figure 3: Data Intake Report

dataset for any class imbalance, however we find an exact 1:1 ratio for each class, highlighting no need for synthetic sampling. A visual distribution of this is shown below:

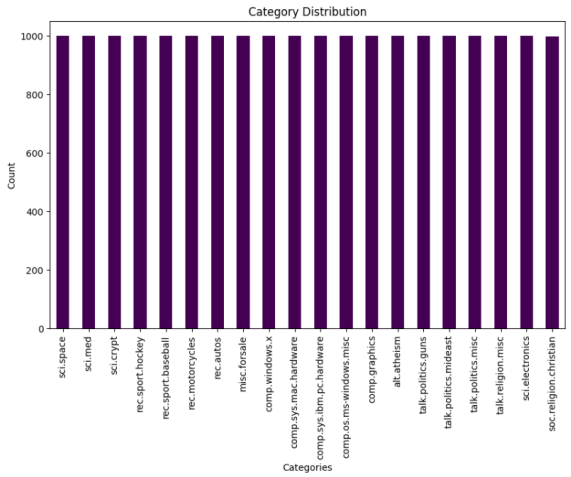

Figure 4: Data Intake Report

5.1 Dataset Pre-Processing

Our pre-processing pipeline contains general techniques initially, such as lower-casing all characters, white-space removal, removing all punctuation except full-stops (for sentence clarity and sentence-tokenisation) and removing stop-words using the *nltk* library. We convert data into numerical vectors representing essential features (Kaushik, 2023). We utilise a Count Vectoriser, which scans through all the documents within the corpus and builds a vocabulary of all unique words, with the vocabulary forming the columns of the resulting matrix. Within each document, the Count Vectoriser scans through all the documents within the corpus and builds a vocabulary of all unique words, and this forms the columns of a

Document-Term Matrix. The result is a

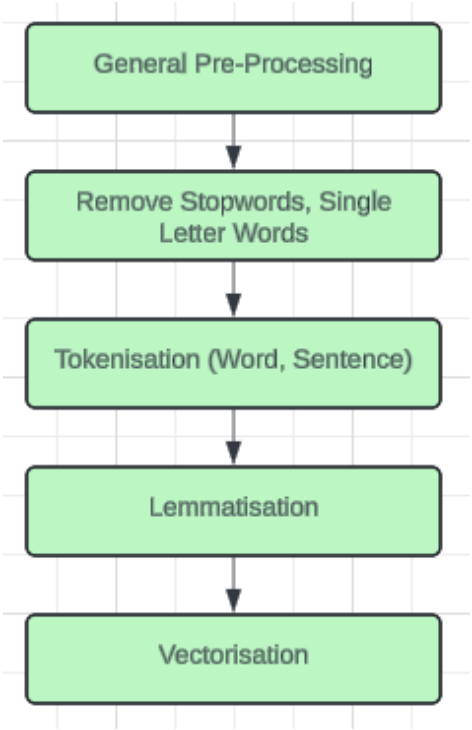

Figure 5: Pre-Processing Pipeline

After General Pre-Processing, we move onto **Word Tokenisation** which is the process by which a large quantity of text is divided into smaller parts called tokens (Johnson, 2024). This breaks the text into smaller parts, to help machines understand human language by creating smaller tokens. This is a key step needed before **Vectorisation** which converts characters into vectors to be analysed and processed in ML/DL Algorithms.

We use **Sentence Tokenisation** as well since it provides a chance to perform text data mining for sentences (GÜBÜR, 2022).

Vectorisation is the ultimate step of our pre-processing pipeline, and it involves converting all our tokenised words into vectors, which can then be fed as input into our ML/DL/Transformer Models.

5.2 Baselines

As our initial baseline model, we use a Dummy Classifier along with a Count Vectorizer and a TF-IDF Transformer, that outputs an accuracy of 0.045. Hence, we expect all of our models to overtake this statistic. We initially test all our models within a pipeline encompassing a Count Vectorizer, followed by a TF-IDF Transformer. An image of this pipeline can be observed below:

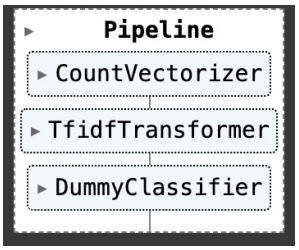

Figure 6: Dummy Pipeline

We decide to train a simple Decision Tree Model, and use this as our Baseline Model for the Machine Learning Section.

A simple 1D Convolutional Neural Network with 3 layers (structure is outlined below) is used as our baseline model to compare our Deep Learning Algorithms too.

6 Results, Error Analysis

For our Machine Learning section, we initially train our models using a Count Vectorizer and a TF-IDF Transformer.

ML Algorithm (Count Vectorizer + TF-IDF Transformer)	Accuracy
Dummy Classifier	0.048
Decision Tree	0.627
Random Forest	0.812
Linear SVC	0.882
Light GBM	0.830
Logistic Regression	0.866
Support Vector Machine (SVM)	0.869
Naïve Bayes	0.864
Gradient Boosting	0.797
XG-Boost	0.812
AdaBoost	0.512
K-Nearest Neighbours (KNN)	0.809
Perceptron	0.789
Bagging Classifier	0.714

Figure 7: Machine Learning Results

From what we can observe, the Dummy Classifier we used as a baseline model was superseded by all the other ML Algorithms in terms of Accuracy. We took a Decision Tree belonging to the Ensemble ML Algorithm family to be our baseline. With a low accuracy of 0.627, this perhaps is due to the large number of classes (20). Decision Tree models are better performing when there is a smaller target subspace, and it would be inefficient

to predict 20 classes from a single Decision tree. This brings us to investigating the Random Forest Model, which is a culmination of Decision Trees that have been trained on different subsets of the data and combined to improve predictive performance and reduce over-fitting. By aggregating the results of multiple decision trees, the Random Forest Model leverages the wisdom of the ensemble, often leading to more accurate and robust predictions compared to individual decision trees. We observe a higher accuracy of 0.812, which is average relative to the other models within Figure 4. The Linear SVC model resulted in being our best-performing model for this dataset, as it is very powerful in high-dimensional feature spaces where data becomes linearly separable. We can observe that the SVM (Support Vector Machine) model came second, at 0.869 with Logistic Regression and Naive Bayes following on. Surprisingly the Logistic Regression model performed well, however this may be due to the effectiveness that it has for high-dimensional text data, as it can estimate the probability of a class based on a linear combination of features, which makes it a strong choice to use with a Count Vectoriser and a TF-IDF Transformer, despite its best use case being Binary Classification. We move on to testing different pipeline structures, shown below:

Pipeline Change	ML Algorithm	Previous Pipeline Accuracy	Current Pipeline Accuracy	Improvement/ Downgrade/ Consistent
Adding a Normaliser	Count Vectorizer + TF-IDF Transformer + Normaliser + LGBM Classifier	0.830	0.830	Consistent
Using the Hashing Vectorizer	Hashing Vectorizer + Normaliser + Linear SVC	0.882	0.869	Downgrade
Customising Hyperparameters on a Vectorizer	Count Vectorizer + TFIDF Transformer + Linear SVC	0.882	0.888	Improvement

Figure 8: Machine Learning Results

The Hashing Vectorizer introduces collisions where different tokens map to the same feature index, leading to the loss of information. This can degrade the performance of the Linear SVC model, since it relies on precise feature representations to find the optimal decision boundary. Adding a normaliser did not change the accuracy, since the normalisation might not significantly affect the gradient boosting model's performance within this specific context, where the Light GBM classifier might already handle the feature scaling internally to some extent. On the other hand, we can customise hyperparameters of the vectoriser which can optimise how the textual data is converted into numerical features. Adjusting parameters within the Count Vectorizer and TF-IDF Transformer can lead to a more informative and relevant feature set. From our three test pipelines which we use to compare to our selected pipeline with different hyper-parameters but the same feature extractors and model, we see a dip in results when the Count Vectorizer adopts a larger n-gram range and the TF-IDF Transformer uses L1 normalisation. A larger n-gram range leads to a large number of features, increasing computational complexity. Furthermore, specific phrases are captured, which may not generalise well leading to over-fitting. L1 normalisation can lead to some features becoming more dominant than others, which can lead to over-fitting.

Model	Accuracy	Loss
Baseline CNN Model	0.842	0.673
Bi-Directional LSTM	0.824	0.956
CNN (Convolutional Neural Network)	0.842	1.177

Figure 9: Deep Learning Results

From our Deep Learning algorithms, surprisingly, our Baseline Model outperformed the other architectures we trained on our testing set. The Baseline CNN model achieved the lowest loss of 0.673 among the three models. This indicates a well-performing model, with a high confidence within its predictions. This simpler, well-optimised architecture only contains one 1D convolution layer, and a dropout rate of 0.5 to handle over-fitting. This simpler architecture helps it generalise unseen data better. The LSTMs are designed for sequential data, capturing long-term dependencies, however spatial hierarchies are more important are not as effective as CNNs; this leads to higher loss if not properly regularised with model struggling to capture correct patterns without overfitting. The CNN model we trained which has a bigram, trigram and fourgram architecture, working on placing the data within 3 different types of kernel sizes results in overfitting, and since we do not inclue many dropout layers, we observe a high instability in training with a significant loss of 1.177, compared with other models. This discrepency suggests issues with the confidence of this model architectures' predictions.

6.0.1 Optimal ML Model

Due to the lack of computational requirements for running GridSearch on all different hyper-parameter combinations of the optimal pipeline (CountVectorizer, TF-IDF Transformer and Linear SVC), we resort to testing three hyper-parameter choices at random. Despite connecting to a T4 GPU High-RAM runtime, which is the fastest version available Google Colab pro-users, the GridSearch algorithm was unable to run, even when we split this into three grids, one per each component.

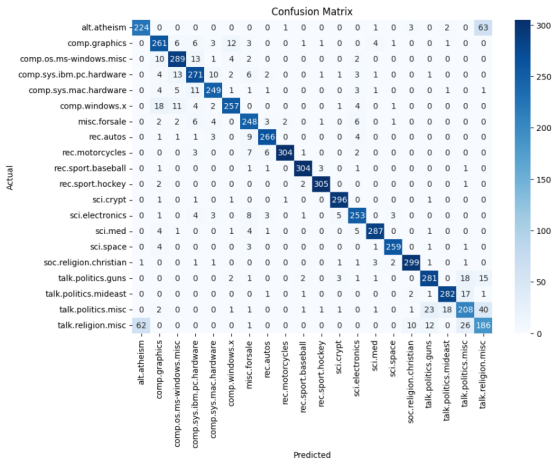

Figure 10: Confusion Matrix for Optimal Linear SVC Pipeline

From our confusion matrix, we can see that the Linear SVC model was able to distinguish between all the different classes accurately. The model struggled to classify news articles which appeared in the "talk.religion.misc" newsgroup and the "alt.atheism" newsgroup.

Our classification report highlights that accuracy, precision, recall and f1-scores for our Linear SVC. We can see that all the metric scores are roughly the same. This consistency across all metrics indicates that the model is well-balanced within its performance, suggesting that the Linear SVC is not only accurate accurate but reliable in identifying relevant documents (high precision) and it has a good trade-off between precision and recall, highlighting this is a robust choice for classifying news articles.

7 Lessons Learned and Conclusions

The main conclusion we have is that the key issue with training models on Text Classification for this

```
Classification Report:
                          precision    recall  f1-score   support

             alt.atheism       0.78      0.76      0.77       294
           comp.graphics       0.83      0.87      0.85       299
 comp.os.ms-windows.misc       0.88      0.90      0.89       321
comp.sys.ibm.pc.hardware       0.84      0.86      0.85       315
   comp.sys.mac.hardware       0.90      0.90      0.90       278
          comp.windows.x       0.91      0.86      0.89       298
            misc.forsale       0.84      0.90      0.87       275
               rec.autos       0.94      0.93      0.93       285
         rec.motorcycles       0.99      0.94      0.96       323
      rec.sport.baseball       0.97      0.97      0.97       312
        rec.sport.hockey       0.98      0.98      0.98       310
               sci.crypt       0.96      0.98      0.97       301
         sci.electronics       0.89      0.90      0.89       281
                 sci.med       0.96      0.94      0.95       304
               sci.space       0.97      0.96      0.97       269
  soc.religion.christian       0.95      0.96      0.96       311
      talk.politics.guns       0.87      0.87      0.87       324
   talk.politics.mideast       0.93      0.92      0.93       305
      talk.politics.misc       0.76      0.70      0.73       298
   talk.religion.misc          0.61      0.63      0.62       297

                accuracy                           0.89      6000
               macro avg       0.89      0.89      0.89      6000
            weighted avg       0.89      0.89      0.89      6000

Accuracy: 0.8882
Precision: 0.8885
Recall: 0.8882
F1 Score: 0.8881
```

Figure 11: Classification Report for Optimal Linear SVC Pipeline

dataset is over-fitting. Articles are unique, and hence complex models which are trained to learn patterns thoroughly will not perform well on this dataset. When we refer to the paper written by (Asim, 2024n), we can see that the mechanism his DCNN model used to rank the words of key importance within the model training itself made a huge difference. Most preferred ML and DL algorithms investigated in this paper, can generalise unseen data well due to the simplicity of the model and the ability to handle non-linear data.

7.0.1 Evaluations/ Conclusions/ Limitations

When answering the question on how Deep Learning Algorithms fare on Document Classification, from the three models we have trained, we found that the simple, baseline CNN model performed the best on unseen data. Of course, there are limitations on this, as we have only investigated a small sample of data, however due to the nature of the dataset having 20 different classes, more complex CNN architectures and Bi-Directional LSTMs have shown to over-fit on training data. We can conclude that for Deep Learning architectures, they have a stronger ability than ML models in identifying patterns, trends and learning tasks, however they are more prone to over-fitting, and require excessive fine tuning.

The Linear SVC model is capable of finding a linear decision boundary within the feature space (often a high-dimensional space) which is

created by TF-IDF or other similar vectorisations of text. The Linear SVC model is able to capture separations between different classes accurately. On the other hand, the Naive Bayes Model assumes that features (words) are conditionally independent given the class label, and this is a strong assumption however it is not true within practice. There can be some inter-dependencies as we observe in the Word Clouds for two newsgroups, write, and article are frequent words which can be in both articles. Despite the TF-IDF Vectoriser ranking words based on their uniqueness to each class, there can be some disparities, especially if there are a lot of words in common between the Newsgroups. Support Vector Machines perform better on non-linearly separable data, which is converted to a higher-dimensional space, which is why the Naive Bayes Model did not outperform the other ML algorithms.

One of our key limitations was the lack of computational resources in iterating through the dataset using *sci-kit learn* GridSearch algorithm, which did not enable us to perform optimisation on our best performing pipeline. The use of Cloud ML services could be a solution for this. Furthermore, we only investigated one dataset, so this paper can only really be applied to the 20 Newsgroup dataset. While the model architectures are justifiable in why they have been proven in text classification, different results may occur for different datasets.

7.0.2 Future Work

Experimenting with different Transformer Architectures, such as different variations of BERT, XL-Net and then models from the GPT Family too. Other considerations further analysis into each Newsgroup, to identify the key phrases/ words which enable algorithms to differentiate test between the different NewsGroups. Some work was done but not completed on investigating Unsupervised Learning Algorithms (LDA - Latent Dirichlet Allocation) and K-Means algorithm. We can delve deeper into the different groups created by these algorithms (setting K = 20 will mirror the number of news groups there are). Perhaps, there are other ways of grouping this data which are better than separating the articles by Newsgroup. Also increasing K, or applying the K-Means algorithm to each newsgroup can bring light to further classification, or another classification step where we can classify the documents in each newsgroup by another factor, which would provide the Company with an enhanced category system for different articles, once the 20 categories become over-filled with an influx of news articles.

We could also investigate further parameter sets on the Naive Bayes, Logistic Regression and Support Vector Machine. Due to time constraints, we only really run optimisation using *Sci-kit Learn* GridSearch feature on the Linear SVC model. However, different hyper-parameter combinations combined with a different Vectorizer/ Transformer could may have resulted in a surprisingly higher accuracy than our optimal Machine Learning Model. Further investigation could also be done to add/ remove additional layers within our CNN, LSTM and Pre-Trained DistilBERT models. We could also investigate more combinations of pipelines with different models, as some feature extractors may work well with specific machine learning models, complementing their algorithm.

Further analysis can be done on specific newsgroups, especially investigating the disparities within alt.atheism and the talk.religion.misc newsgroups, to see why miss classifications occur.

8 Word Count Table

Section	Word Count
Problem Statement and Motivation	297
Research Hypothesis	283
Related Work and Background	763
Approach and Methodology	823
Dataset	511
Results, Error Analysis	1032
Lessons Learned and Conclusions	281

Figure 12: Word Count

References

Muhammad Asim. 2024n. A robust hybrid approach for textual document classification. *National Center of Artificial Intelligence*.

Baeldung. 2024. Prevent the vanishing gradient problem with lstm. Accessed: 2024-08-04.

Aniruddha Bhandari. 2024. Feature scaling: Normalization and standardization. *Analytics Vidhya*. Accessed: 2024-07-07.

J Sree Devi. 2020. Newspaper article classification using machine learning techniques. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 9(5).

Koray Tuğberk GÜBÜR. 2022. Nltk tokenize: How to tokenize words and sentences with nltk? Accessed: 2024-07-07.

IBM. 2023. What are recurrent neural networks? Accessed: 2024-08-04.

Daniel Johnson. 2024. Nltk tokenize: Words and sentences tokenizer with example. Accessed: 2024-07-07.

Vikas Kaushik. 2023. Understanding vectorization: Applications, benefits, and future trends. *Medium*.

Donghwa Kim, Deokseong Seo, Suhyoun Cho, and Pilsung Kang. 2019. Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec. *Information sciences*, 477:15–29.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE.

Sameer Pradhan, Julia Bonn, Skatje Myers, Kathryn Conger, Tim O'gorman, James Gung, Kristin Wrightbettner, and Martha Palmer. 2022. PropBank comes of Age—Larger, smarter, and more diverse. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 278–288, Seattle, Washington. Association for Computational Linguistics.

Henk van der Duim. 2023. Why datasets are crucial to data science: the key to informed decisions. *HackerNoon*. Accessed: 2024-07-07.