Report on Text Classification using NLP

by Mohammedzaki Shaikh

1 Introduction

Research question: "What are the effective machine learning approaches and techniques for text classification of the 20 Newsgroups dataset?" We will investigate different feature selection methods and combine them with different machine learning algorithms to understand the effectiveness of each combination of methods.

2 Literature Review

Text classification can be defined as the allocation of one or more categories from a predefined set of labels to a document, according to its content (Dumais et al, 1998; Yang et al, 1999; Ruiz et al, 2002). The automated categorization/classification of texts into predefined categories has become very popular, due to the increased digital document availability and the requirement to organize them (Sebastiani, F. 2002).

The primary approach to this is based upon machine learning techniques: an inductive process which automatically builds a classifier by learning, category characteristics, from pre-classified document sets. This approach has greater advantages over the knowledge engineering approach as it is effective (Joachims, T. 1998). To address our research question, here are some points which we will focus on in upcoming sections:

Exploratory Data Analysis (EDA): Before building a classification model, it's necessary to conduct a shallow exploration (text analysis and visualization) to understand the data storage mode. EDA plays a crucial role in text classification problems to help uncover patterns and insights in the data, guiding the development of effective models (Bishop, 2021). The insightful conclusions obtained through this will also determine the logic of building the model to a certain extent.

Pre-processing: Pre-processing text data can help improve the performance of text classification algorithms. Through data observation in the previous step, some texts contain non-English content and acronyms. If these words are not processed in advance, the analysis accuracy is affected resulting in false negative and false positive (Baeza-Yates, R., & Ribeiro-Neto, B. 2011). Secondly, it's necessary to remove non-key information within text, such as: headers, footers, quotes, numbers in the body text, special symbols, etc (Jurafsky, D., & Martin, J. H. 2019).

Feature Extraction and Selection: Necessary step since machine learning models require numerical values for their input. Feature extraction for text classification could involve extracting variables such as article length, proportion of nouns or sentence length. During EDA, we discovered that most articles were a similar length and had a similar proportion of word types. Thus, we have used word-based feature extraction methods whereby the words within each group of articles are analysed and transformed into a feature set to be usable by a classifier. Two techniques were considered:

- The Bag of Words (BoW): works by creating a vocabulary list of all words in the corpus, and then for each document, creating a vector that represents the word frequency in the vocabulary list. **(**Manning, C. D., Raghavan, P., & Schütze, H. (2008).

- The Term Frequency Inverse Document Frequency (TF-IDF) model varies from the BoW model as it considers the word frequency in a document, as well as the inverse document frequency. This means that the TF-IDF model is more likely to identify the important words in a document whilst removing any potential bias. (Salton, G., & Buckley, C. 1988)
- Word embeddings: is a natural language processing technique mapping words or phrases to a vector of numerical values. Word embeddings are created using neural network-based models.

Subsequent or concurrent feature selection methods can be used to improve classification accuracy of a machine learning model by automatically or manually selecting features which could contribute most to the classification (Pintas et al, 2021). Feature selection often increases classification accuracy by eliminating noise features (Amruthjithraj, 2020).

Machine learning model

There are many different machine learning algorithms that can be used for text classification including Naïve Bayes, Support Vector Machines (SVM) and K-Nearest neighbours (KNN) as well as Deep Learning techniques.

Naïve Bayes: A commonly used text classifier and is a focus of research in text classification (Rennie, 2014). It copes with classification of a large dataset in a relatively short amount of time and results are easy to interpret.

Support Vector Machines: An SVM learns a decision boundary/hyperplane between different classes in the dataset. The goal is to identify the best hyperplane that separates two classes by maximizing the margin.

K-Nearest Neighbours: Aims to predict the label of a new data point based on the labels of its "nearest" neighbours in the training dataset. The value of K determines the number of nearest neighbours that are considered in the prediction of the new label, and the predicted label is either the mode or the mean (for regression) of the labels of the K nearest neighbours.

Decision Tree: In a decision tree, each node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a class label or a predicted value. The tree algorithm partitions data into subsets based on feature values and then selects the best feature to separate data into different classes based on a criterion such as entropy or Gini impurity. The feature with the greatest criterion reduction is selected to split data into two or more subsets. This is repeated until a stopping criterion is met.

Ensemble methods: Combining multiple machine learning models using ensemble methods such as bagging and boosting can improve performance. Here Uniform Weighting method has been used.

Artificial Neural Network (ANN): A typical neural network has multiple (potentially millions) of artificial neurons arranged in a series of layers, each of which connects to the layers on either side. ANNs frequently out-perform other machine learning techniques (Geron, 2019).

Convolutional Neural Networks (CNNs): originally invented for computer vision, have been shown to achieve strong performance on text classification tasks (Bai et al., 2018; Kalchbrenner et al., 2014; Wang et al., 2015). Yoon Kim (2014) describes how you can use convolutional layers to find patterns in sequences of word embeddings and create an effective text classifier using a CNN approach. However, this method requires substantial computer processing power and time.

It is difficult to predict which method will be 'the best' without model testing and result analysing.

Parameter Tuning

Hyperparameter optimization: Many machine learning algorithms and deep learning models have hyperparameters that need to be tuned for optimal performance. Investigating different techniques for hyperparameter optimization, including random search (where hyperparameters are randomly selected from a predefined search space), grid search (where gridded predefined hyperparameters and algorithms try various hyperparameter combinations) and Bayesian optimization. This can help identify the best hyperparameters for each algorithm and model.

Evaluation metrics: Different evaluation metrics such as accuracy, precision, recall, and F1-score can be used to evaluate the performance of text categorization algorithms on the dataset. For an equally distributed text classification task, like ours, mean accuracy would be a suitable measure of performance, however it would also be useful to inspect the confusion matrix to understand if there are any categories which are not allocated as accurately as others. (G Handelman, et al, 2019)

Cross-validation: To ensure that the results obtained from text categorization experiments are reliable, it is important to use cross-validation techniques such as k-fold cross-validation. Investigating the effectiveness of different cross-validation techniques on the dataset can help ensure the validity of the results.

Overall, investigating these different areas can provide valuable insights into the most effective approaches and techniques for text categorization using the dataset.

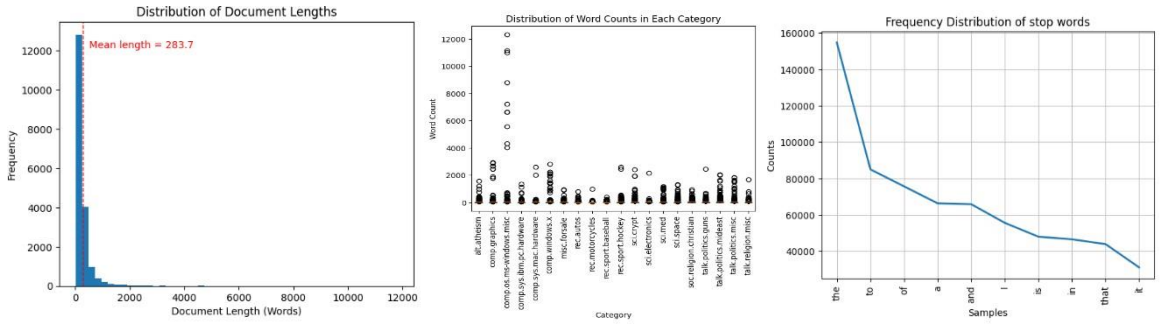
3 Description of the Dataset

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. This includes topics, such as politics, religion, sports etc. The dataset has been divided into training and test data (data collected after a particular date).

4 Methodology

4.1 Exploratory Data Analysis

In this project, EDA was used to gain a better understanding of the text data used for classification. By performing EDA, the researcher can identify the most common words and phrases, the distribution of word frequencies, and the relationship between different variables. **4.1.1 Distribution**



Several correlation distributions have been plotted to understand text data. These distributions are related to document length, category word count, and text data quality. The graph showing the distribution of document length shows that the average document length is 283, and there are cases where the number of words is very large or very small.

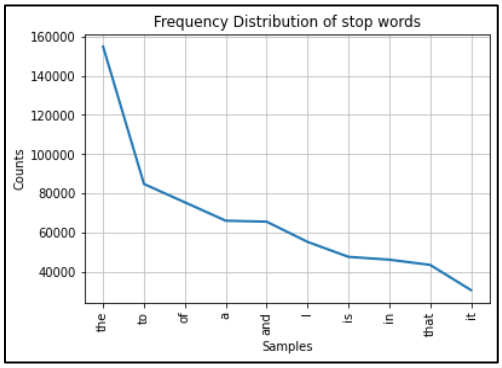
The box plot shows that the 'comp.os.ms_windows.misc' category contains multiple documents with word counts exceeding the 4000 words mark whereas no other category has this. However, each

category does have a very similar mean number of words per document, as seen by the large cluster of points for each class, despite the vast changes in max word count across all categories.

We also check the text data quality we are given through a distribution which shows the frequency of slang terms. The bar plot shows the top 20 slang words in our dataset. As we can see from the bar plot, 'btw' was written by users considerably more than any other slang word however, this abbreviation still only appeared less than 500 times. This is quite a low frequency of unprofessional words hence the quality of the text data is good.

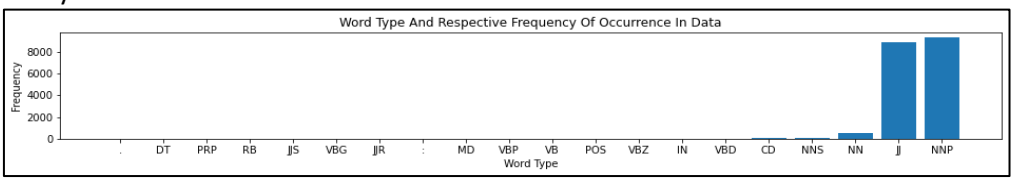
4.1.2 Stop words

Stop words don't usually provide much insight into the data. Hence, these words can be filtered out to aid analysis and processing. Use the Natural Language Toolkit (NLTK) to find the frequency of each specific stop word. The frequency distribution plot shows the top ten most frequently used stop words, where the word "the" is heavily used. However, even the tenth most used stop word "it" appeared at a very high frequency, reinforcing the decision to remove all stop words from the text, optimising the efficiency of the code when processing the data.



4.1.3 Word Type and Frequency

The NLTK, POS tagging method was used to identify specific word categories in the dataset, whilst the bar chart shows how frequently those words appear. Proper nouns (NNP) and adjectives (JJ) are the two-word kinds that appear most frequently, with high values being represented by bars that tower over the other word types. In contrast, all other word kinds are displayed at negligible or low levels. The horizontal axis of the graph shows the various word types, while the vertical axis of the chart shows the frequency of occurrence.



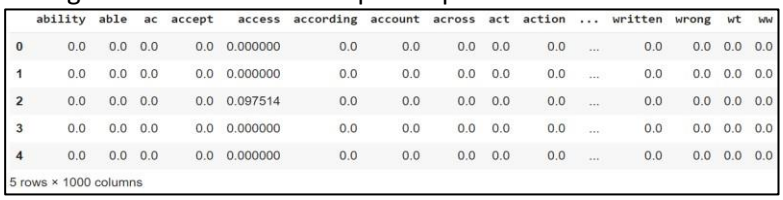
4.2 Pre-Processing

The pre-processing function is applied to training and testing data to reduce noise by removing redundant information, helping to improve the text classification algorithms performance and ensure data consistency and comparability. The main idea of processing is to normalize text by converting it into a standard format, and to tokenize it by breaking it down into smaller units. We define preprocessing functions to remove non-alphabetic characters, convert text to lowercase, tokenize text, remove stop words, remove numbers, lemmatize and concatenate them back to a string. Considering the impact of word stem extraction on the classification accuracy of information (especially news information), this method is not adopted. Through experimentation, it was found that removing outliers did not have a significant impact on the final model evaluation results. Therefore, they are preserved.

4.3 Feature Extraction

1. Term Frequency-Inverse Document Frequency (TF-IDF)

The figure below shows a sample output from the TF-IDF feature extraction method.



	ability	able	ac	accept	access	according	account	across	act	action	...	written	wrong	wt	ww
0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.097514	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 1000 columns

2. Bag of words: represents the text data as a set of features which can be used for text classification. It's particularly useful because the ordering of words isn't important, and for our text classification our focus is on word frequency within a document or a collection of documents. When checking the feature matrix shape, we can see that there are 114339 new features created – representing each word in the set of documents.

OUTPUT: Shape of training feature matrix: (11314, 114339) and Shape of test feature matrix: (7532, 114339)

3. Word embeddings: In our case we have used Word2Vec as it captures syntactic and semantic relationships between words aiding in our text classification.

4.4 Methodology and experimental settings

We have trained and tested various machine learning models for this classification problem. There are 11,314 news articles in the training/development set and 7532 articles in the test set. Cross validation techniques are utilised throughout with random search being favoured. Accuracy, precision, recall and F1 scores will be compared for each model as well as inspection of the confusion matrix and learning curves. We have tried simpler baseline models (model 2,3 and part of 6) an ensemble method (model 4) and several neural network approaches (model 1, 5 and 6).

Model 1: Neural Network (NN) – using Keras Sequential model:	Define the input dimensions from pre-processed dimensions training set. This node passes on the features of features of the dataset to the hidden layer. We have used the **RELU activation** function for our hidden layers of the NN since it's much faster to compute than the sigmoid and Tanh function. We have used the **SoftMax activation** function for the output layer since it can handle multiple classes. Loss Functions: these are the target the NN is optimised for during training. Since this is a classification problem, we have used cross-entropy as the loss function. We have used **sparse_categorical_crossentropy** since the target variable is a label (keras documentation ref). Validation set: To avoid overfitting the data, we have used a validation set for optimising the loss function. We have chosen to use 20% of the training set for this by using validation_split=0.2 during model fitting. Optimizer: We have used **Adaptive Moment Estimation (ADAM)** as our optimizer for the loss function. This was selected due to its favourable performance demonstrated in the literature (Kingma et al, 2014) and the high

	speed of computing. We have used the **EarlyStopping** call-back in Keras to stop training once validation accuracy is maximised. We have also used the **ModelCheckpoint** call-back to save the best model observed during training. Early stopping is monitoring loss on the validation dataset and model checkpoint is saving models based on accuracy.

Model 2 - Naïve Bayes with pipeline and random search hyperparameter turning	We have used a Naïve Bayes classifier, before training this model, we used random_searchCV for hyperparameter tuning to tune both the parameters for the vectorizer and the parameter for Naïve Bayes. This also allows us to use validation set (kfold = 5)to avoid over fitting of the data.
Model 3: K-Nearest Neighbours	A basic KNN model was first created and run, using the default values for all hyperparameters, to serve as a baseline. A grid search was then performed with multiple possible values for hyperparameters, and a new KNN model was created using the best set of parameters identified. • N-neighbours – The number of neighbours to be considered in the prediction of the new label. For this, a range of 1-20 was provided for the hyperparameter grid search. • Weights – This specifies how to weight the contributions of the neighbours to the prediction. For this, both the "uniform" and "distance" weight options were considered. • Algorithm – This specifies the algorithm used to compute the nearest neighbours. For this, "auto," "ball-tree," "kd_tree," and "brute" were provided for the hyperparameter grid search.
Model 4: Decision Tree model	A basic decision tree model was first created and run, using the default values for all hyperparameters, to serve as a baseline. A grid search was then performed with multiple possible values for hyperparameters, and a new decision tree model was created using the best set of parameters identified. • Max-depth – This is the maximum depth of the decision tree, and if unspecified, nodes are expanded until all leaves are pure or until all leaves contain less than specified minimum samples for split. • Minimum leaf samples – This is the minimum number of samples required to be at a leaf node. • Criterion – The function used to measure the quality of a split. It can be either 'gini' or 'entropy'. • Splitter – The strategy used to choose the split at each node. The options are "best" to choose the best split or "random" to choose the best random split.
Model 5: CNN using keras library and hypertuning through RandomSearch:	The maximum sequence length is set to 100 and then pad the sequences of word embeddings to a fixed length. Then the code creates a simple CNN model with two convolutional layers, each followed by a max pooling layer, a flatten layer, two dense layers, and a dropout layer. The model takes as input the sequences of embeddings padded to the fixed length and outputs a probability distribution over the 20 possible classes for our data. Finally, the model is trained on the padded training data and one-hot encoded training labels for 10 epochs with a batch size of 32. The validation data is set to the padded testing data and one-hot encoded testing labels. The RandomSearch class from Keras Tuner performs a random search for the best model hyperparameters. The RandomSearch object takes as input the
	build_model function, the hyperparameters to search over, the number of trials to run, and the objective to optimize (the validation accuracy).

Model 6: Ensemble Model (CNN and SVM)	CNN utilises two convolutional layers with each layer followed by a max pooling layer. The number of filters of the convolutional layers gradually increases from 32 to 64. A dropout layer is added preventing the model from overfitting. Finally, the softmax activation function adds output layer, containing 20 neurons (corresponding to 20 categories). SVM uses a linear kernel function, with the parameter C set to 1, enabling probability estimation. By weighting the prediction results of CNN and SVM models, the final prediction result is obtained. Combining these two models improves the model generalization ability as CNN captures local features of sequence data and SVM deals with sparse data, linearly separable problems and non-linear problems. As the two models have different limitations, a simple ensemble method is used to obtain the predicted probabilities weighted average of the models. 1. Use 5-fold cross-validation, which divides the training data set into 5 parts, use one of them as the verification set, and the rest as the training set. Hierarchical cross-validation ensures that the proportion of samples of each category remains the same in the training set and validation set. 2. Use the kfold.split() method to split the dataset. In each iteration, the data set will be divided into training and validation set according to the index obtained by stratified cross-validation. 3. Use divided data to train the CNN and SVM models. The weighted average of the models predicted probabilities is calculated to calculate the ensemble predicted probability (ensemble_prob). In this example, the weights are equal, i.e., each model has a predicted probability weight of 0.5. 4. The most probable class (ensemble_pred) is selected according to the ensemble predicted probabilities, i.e., the index along axis 1 that takes the maximum value.

5 Results

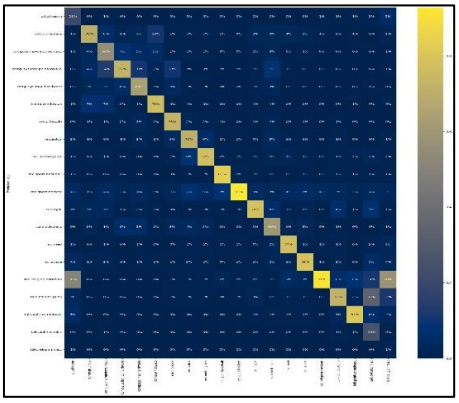
5.1 Evaluation Metrics

Model	Accuracy	Precision	Recall	F1 Score
Naïve Bayes	**0.68**	**0.70**	**0.66**	**0.68**
K-Nearest Neighbours	**0.50**	**0.48**	**0.45**	**0.46**
Decision Tree Model(Pre-tuning)	**0.29(0.46)**	**0.47**	**0.46**	**0.46**
SVM	**0.48**	**-**	**-**	**-**
CNN & SVM	**0.91**	**0.92**	**0.91**	**0.91**
CNN (with word embeddings)	**0.55**	**0.88**	**0.59**	**0.70**

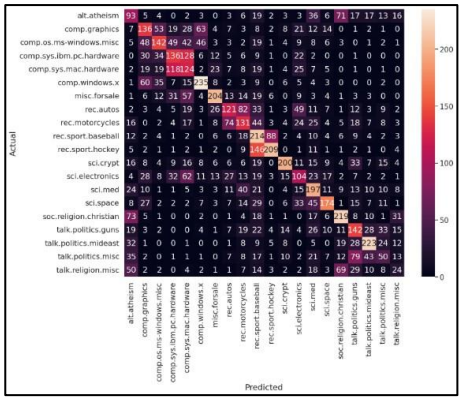
From these metrics we can determine that the CNN and SVM ensemble method had the best overall performance with the highest scores in all analysed metrics. The model correctly predicted the text category 91% of the time with high precision and recall factors. The more basic CNN model had the next highest accuracy metric but had significantly lower Recall and F1 values in comparison to ensemble model. The Naïve Bayes model performed the next worst providing the lowest values for each evaluation metric. These metrics do indicate that every model showed a level of success when classifying, as when considering a multi-class classification problem of this scope a randomised baseline model would establish an accuracy of 5%.

5.2 Confusion Matrix

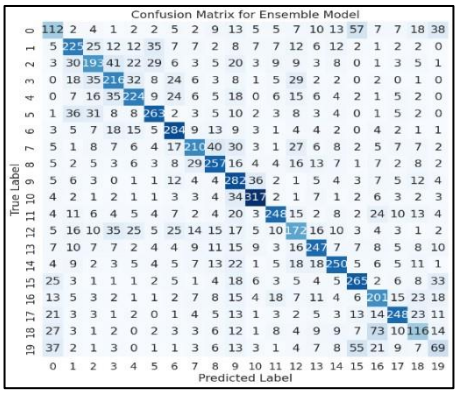
Confusion Matrices evaluate the performance of models by showing which classes have been predicted correctly and which get confused. From the matrix, we can build a heatmap which visualises the model accuracy and allows conclusions to be drawn.



Naïve Bayes - Naivety is due to the model assuming independence between the classes. In our case this can be relatively useful for overall accuracy however, as illustrated in the heatmap, certain classes are rarely predicted. For example, the 'talk.religion.misc' category has an extremely low accuracy due to this independency assumption. We know this as a high proportion of the documents from the 'talk.religion.misc' class have been predicted into 'soc.religion.christian', clearly generalising religious terms to some extent through the model's naivety. Overall, the Naïve Bayes model does predict accurately except in the occurrences of miscellaneous topics, which the model can sometimes overlook.

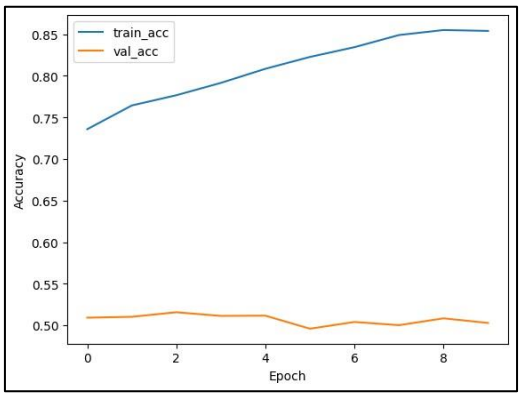


CNN hypertuned using RandomSearch - The heatmap, like previous models, depicts how this model confuses many classes that are quite similar. For example, baseball and hockey clearly had a lower success rate with correct predictions compared to other classes. The model was able to predict almost every class, albeit with a slightly lower overall accuracy compared to some other models. One class that was particularly hard to predict was 'talk.religion.misc' possibly due to its large overlap with 'alt.atheism' and 'soc.religion.christian'. The CNN model which has been hypertuned with RandomSearch is evidently not as accurate overall in comparison.



CNN and SVM Ensemble model - Due to being an ensemble model combining these particular models, it's expected that the accuracy would be higher than that of single models. As seen in the heatmap, this is the case. Other heatmaps from different models can be seen to have sections of 'hotness' around two or three categories, caused by class confusions. This heatmap does not appear to have such a drastic issue in comparison. In fact, unlike other confusion matrices, the ensemble model here can somewhat accurately predict the 'talk.religion.misc' class; something which has been difficult to achieve. As seen by the metrics table, the confusion matrix of the ensemble model proves the high accuracy and precision. The heatmap contains a distinctive diagonal line which is very hot while the rest of the graph is extremely cold, therefore showing how the model hasn't had much confusion with any classes.

5.3 Learning Curve



CNN – The optimisation training and validation curves appear to follow a similar pattern and demonstrate that the plot of training loss decreases to a point of stability and the plot of validation loss decreases to a point of stability and has a small gap with the training loss, indicating that overall, the model is well fit. The performance curve also shows the model is fully trained plateauing at its maximum accuracy of %70.

CNN and SVM Ensemble Model - The graph shows the model is not underfit as the training loss doesn't remain flat regardless of training and the training loss doesn't continue to decrease until the end of training. The graph also represents that the model is not overfit as the training as the training loss does not continue to decrease with experience and the plot of validation does not decrease to a point and begin increasing again. This means that the model has a good fit. The accuracy plots demonstrate the model's strong performance and ability to learn well plateauing at its maximum accuracy of 91%.

The evaluation of all the learning curves insights that both the training and validation datasets appear to be well representative and capture the correct statistical characteristics. Both models show they have a good fit, however the ensemble model's maximum performance capacity is higher than that of the CNN model by itself.

6 Evaluation

The EDA that was initially conducted gave great insight into multiple traits that helped build the models. Exploring slang and stop words proved to be useful when it came to pre-processing the raw data. As seen in the metrics table, the lowest f-1 score was around 50% which is quite high considering the complexity of the dataset. Given this, the ensemble model provided an extremely high score of about 90%, clearly showing success in the decision to combine two common models.

As previously mentioned, the dataset contained many classes that were quite similar in topic. All the models struggled to classify documents into these classes, albeit with varying degree. When attempting to hypertune the first CNN model with RandomSearch, the model received much lower accuracy scores than previously. The hypertuning appeared to cause significant generalisation in the data and hence validation accuracy was low in comparison training accuracy.

In Conclusion, the models performed effectively given the scope of the multi-class classification problem and all models were able to classify to a level that is considered better than random selection. There were clear adversities faced within the dataset provided, through research and development these were identified and combated through effective exploration, processing and model production. The overall results demonstrated that the CNN & SVM ensemble model performed the best out of each machine learning technique considered.

If we were to do the project again, we believe that we could be more strategic and methodical with model selection and tuning. For example – we could try the Naïve Bayes model (best simpler model) with the CNN ensemble.

References:

1. Baeza-Yates, R., & Ribeiro-Neto, B. (2011). Modern Information Retrieval: The Concepts and Technology behind Search (2nd ed.). Addison-Wesley Professional.
2. Bishop, C. M. (2021). Pattern Recognition and Machine Learning. Springer.
3. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: Analyzing text with the natural language toolkit. O'Reilly Media, Inc.
4. Das, D. (2018). Text classification and naive bayes. Journal of Emerging Trends in Computing and Information Sciences, 9(5), 243-252.
5. Deng, X., Li, Y., Weng, J., & Zhang, J. (2019). Feature selection for text classification: A review.
6. Esmaeilpour, M., Nematbakhsh, M. A., Ghaeini, H. R., & Zamanifar, M. (2016). A comparative study of feature selection and classification algorithms for sentiment analysis. Journal of Ambient Intelligence and Humanized Computing, 7(3), 417-430.
7. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Ed.). Farnham: O'Reilly.
8. Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.). Pearson Education.
9. Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing. Pearson.
10. Kingma, P.D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization.
11. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.
12. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
13. Nielsen, M. (2019) Neural Networks and Deep Learning. Accessed: Neural networks and deep learning.
14. Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513-523.
15. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1-47.
16. G Handelman, H Kok, R Chandra, A Razavi, S Huang, M Brooks, M Lee, and H Asadi(2019). Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods, American Journal of Roentgenology 2019 212:1, 38-43.