1. Introduction

1.1 Preprocessing and Visualisations

The dataset contains a train and test set, with the first being labelled by the helpfulness of the review and the second not. In the train data, 76% of the reviews (8214) are useful and 24% of them are not so useful (2541), based on the users. The train set is going to be used in order to classify the label of the reviews on the test set and classify if they are helpful or not.

The pre-processing of both train and test datasets consists of removing special characters, such as slashes, and cleaning the stopwords, which are presented below by category of helpfulness on the train dataset. (Figure 1)

In addition to this, in order to proceed with the Machine Learning models, the transformation of the reviews into string format is required. Finally, the preprocessing of the data is done with the transformation of the labels of the helpfulness_cat column into a numeric format, from “0.0” and “1.0” to 0 and 1.

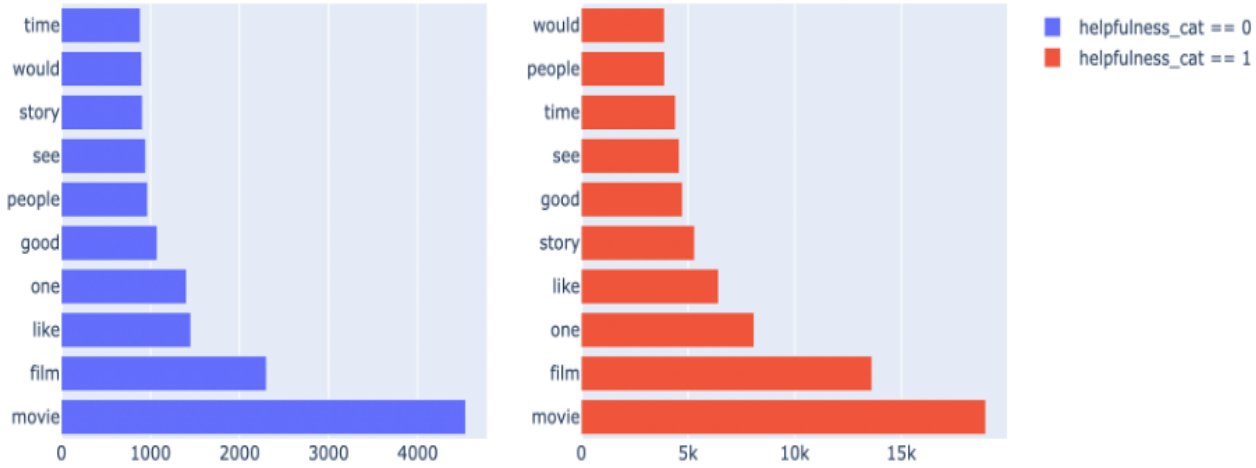


Figure 1. Stopwords count by category.

1.2 Wordcloud

The Word cloud is a data visualisation technique used for representing text data in which the size of each word indicates its frequency or importance. The more common words from the text analysed the larger they appeared in the plot generated. In the Word cloud generated from the train dataset, the most common words are “movie”, “film”, “story”, “good”, “character”, “people” and “scene”, words that the users used in order to comment about the characteristics of the film, such as the plot, the story or the characters. (Figure 2)

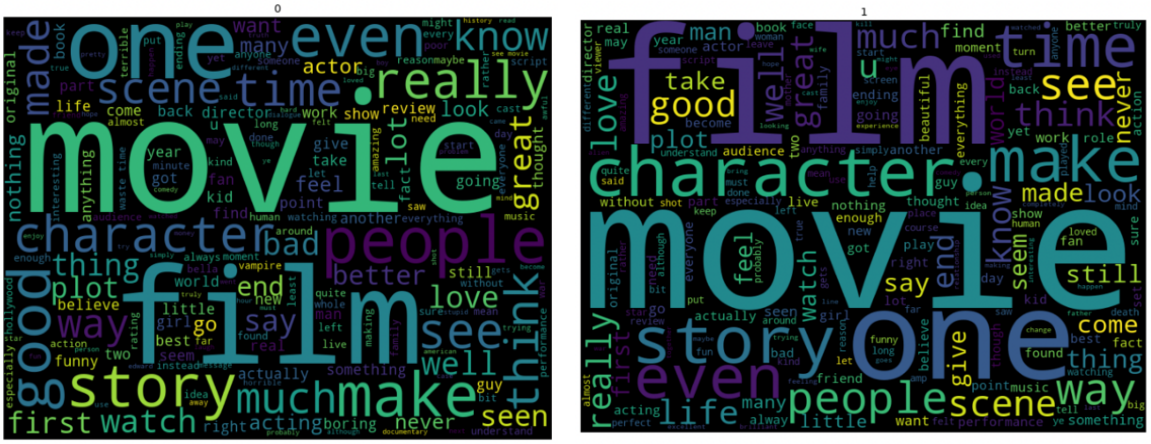


Figure 2. Word cloud of reviews labelled as useful and not useful.

2. NLP tools

2.1 Bag of words

The bag-of-words is an NLP technique of text modelling. BoW is turning the arbitrary text into fixed-length-vectors, by counting how many times each word appears. The BoW is simple to understand and implement and is very successful in problems such as language modelling and document classification. The steps followed to complete the BoW model are to create and design the vocabulary, create the document vectors and apply multidimensional scaling in order to reduce the dimensionality of the data. Below is the transformed data table and its visualisation. (Figure 3 , Figure 4)

	x	y
0	9.599201	1.303573
1	17.370481	-13.459288
2	-5.685441	-4.642546
3	-6.939367	20.089469
4	3.119391	3.314441

Figure 3. The transformed data frame

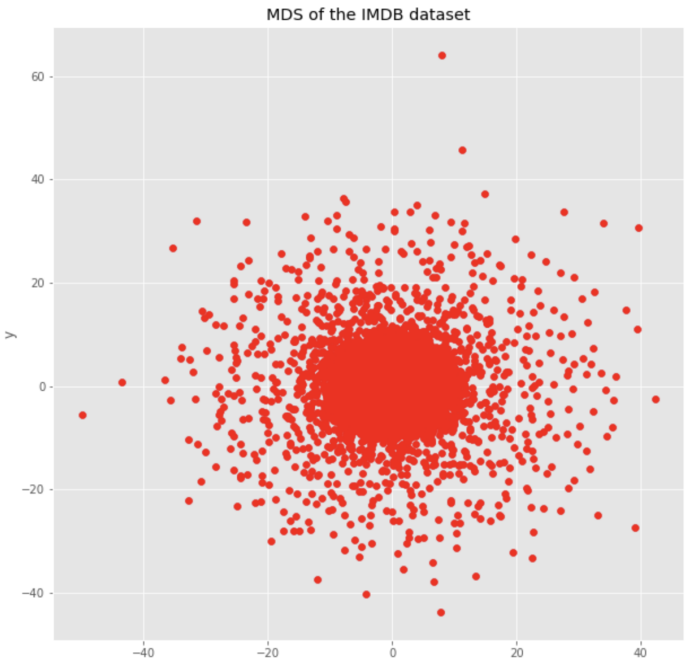


Figure 4. Visualisation of the transformed data frame

2.2 TF-IDF and N-Grams

Term Frequency-Inverse Document frequency is a statistical measure and is used in order to identify the importance of a word to the document. By using TF-IDF, words are given weight, measured by relevance, rather than frequency. TF-IDF is conducted by two statistics, the Term Frequency, which is the number of times a word appears in a given document and the Inverse Document Frequency, which defines the frequency a word appears in more documents, the less that word is as a single.

```
['−PRON−', 'AA', 'AAA', 'AAAAAAAAAAGGGGHHHHHHHHHH', 'ABC', 'AC', 'ACD', 'ACLU', 'ADHD', 'AFI', 'AGE', 'AHEADOK',
'AHEADSo', 'AHS', 'AI', 'AIDS', 'AIRPORT', 'AJ', 'AL', 'ALERT', 'ALEX', 'ALIEN', 'ALLY', 'ALPACAS', 'ALS', 'ALYOSHA',
'AMC', 'AMY', 'ANANYA', 'ANGEL', 'ANNA', 'ANNIE', 'ANURAG', 'AOTC', 'ARC', 'ASAP', 'AUGUST', 'AWESOME', 'AWOL',
'Aalia', 'Aaliya', 'Aamir', 'Aamirs', 'Aamitabh', 'Aang', 'Aannd', 'Aaron', 'Aarons', 'Aayiram', 'Abarams', 'Abbey',
'Abbie', 'Abby', 'Abdel', 'Abdelghafour', 'Abdul', 'Abe', 'Abeds', 'Abel', 'Abhay', 'Abhishek', 'Abhisheks',
'Abhiskek', 'Abi', 'Abiding', 'Abigail', 'Abignale', 'Abis', 'Abkhaz', 'Abkhazia', 'Aboriginals', 'Aboriginies',
'Abott', 'Abou', 'Abraham', 'Abrahams', 'Abrahamson', 'Abrahamsons', 'Abrams', 'Abraxis', 'Abre', 'Abu', 'Abudabhi',
'Aby', 'Acadamy', 'Academy', 'Acclaim', 'Accountant', 'Accuracy', 'Ace', 'Acevedo', 'Ach', 'Acharya', 'Acheman',
'Achievement', 'Ackerman', 'Ackermans', 'Ackroyds', 'Acosta', 'Act', 'Acting', 'Action', 'Activity', 'Actor',
'ActorClint', 'ActorKathy', 'Actore', 'Actors', 'Actress', 'ActressSam', 'Adam', 'Adams', 'Adamsleigh', 'Adamson',
'Adamss', 'Adaptation', 'Addai', 'Addams', 'Adelaide', 'Adelaides', 'Adele', 'Adina', 'Adja', 'Adlen', 'Adler',
'Administration', 'Admiral', 'Adolf', 'Adolphe', 'Adolphs', 'Adonis', 'Adrenaline', 'Adrian', 'Adrien', 'Adult',
'Adventure', 'Adventurer', 'Advice', 'Adzovic', 'Aeschylus', 'Affairs', 'Affif', 'Affleck', 'Afflecks', 'Afganistan',
'Afghan', 'Afghanistan', 'Afghans', 'Afleck', 'Afraid', 'Africa', 'Africaaners', 'African', 'Africans', 'Afro',
'Afterlife', 'Afternoon', 'AfterthoughtsUnderstandably', 'Agatha', 'Agathas', 'Agda', 'Age', 'Agency', 'Agent',
'Agents', 'Ages', 'Agnes', 'Agnihotrys', 'Agora', 'Agrabah', 'Agresti', 'Agrestis', 'Aguirresarobe', 'Ah', 'Ahmed',
'Ahmeda', 'Ahmeds', 'Ahna', 'Ai', 'Aickman', 'Aid', 'Aidan', 'Aids', 'Aiellos', 'Aijla', 'Air', 'Airbender', 'Aires',
'Airmen', 'Airplane', 'Airport', 'Airstation', 'Aishwairya', 'Aishworya', 'Ajay', 'Ajit', 'Akbar', 'Akerman', 'Akim',
'Akin', 'Akira', 'Akita', 'Akkad', 'Akki', 'Akshay', 'Akshya', 'Aksongur', 'Akua', 'Al', 'Alabama', 'Aladdin',
```

Figure 5. TF-IDF vocabulary

2.3. Document to Vector

In order to understand Doc2Vec, the knowledge of Word2Vec is essential. Word2Vec is a well-known concept, used to generate representation vectors out of words based

on distributional hypotheses - suggesting that semantically similar words will be distributionally similar and occur more frequently in similar linguistic contexts. The word embeddings of similar words will also represent those words closely in the vector space. In general, when building a model using words, labelling or one-hot encoding them is a plausible way to do it. However, when using such encoding, the meaning of the words is lost, and with Word2Vec, a numeric representation for each word, that captures the relationship of the word, is created.

Doc2Ve, an extension to Word2Vec, is an NLP tool for representing documents as vectors and its goal is to create a numeric representation of a document, regardless of its length.

	x	y
0	0.598079	-0.429047
1	0.566558	1.192479
2	0.361428	-0.944624
3	0.489220	0.229809
4	0.651549	-0.456743

Figure 6. The transformed data frame

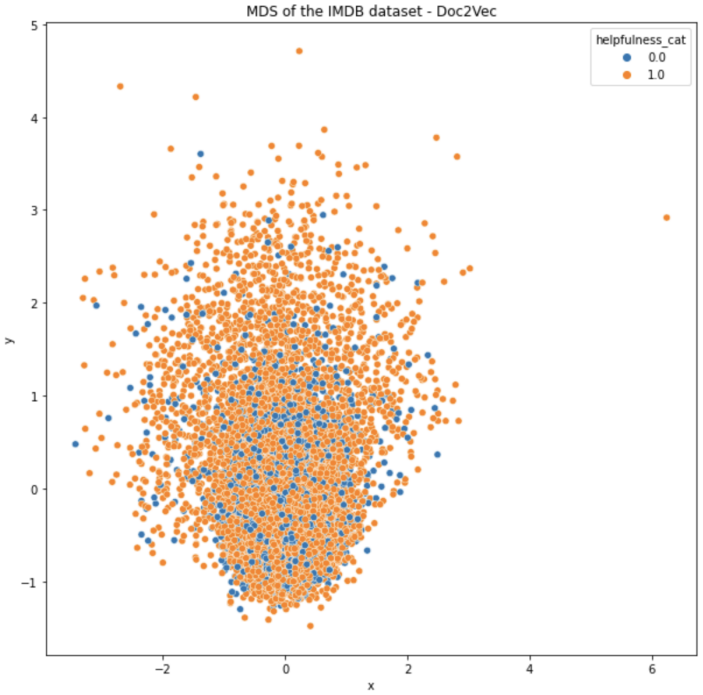


Figure 7. Visualisation of the transformed data frame

2.3. Topic Modelling

Latent Dirichlet Allocation represents documents as a collection of topics, which are further composed of a collection of words. LDA, a statistical modelling tool assesses what all abstract topics are being discussed in a set of documents and allocates a probability of each document to each topic. Topic Modelling can be used as a tool to derive features for documents, in this case for each IMDB user review. The reviews are explained as probabilities spanning across k number of topics covered in the overall corpus of the reviews. As explained by Li et al 2016, the features are not necessarily high dimensional.

The training set was used to train 200 different topic models ranging from topic sizes between 1 and 200. During the first iteration of LDA the optimal topic size was 198 and the extremely frequently occurring words were film, movie, and character. These words were then put in the custom stopwords before re-running the LDA in order to eliminate the possibility of topics to create affiliations with these words (Schofield et al 2017). Since they are movie reviews, the topics will revolve around such words. Bigrams and trigrams were also identified and incorporated in the corpora. The number of topics that rendered the best (lowest) coherence score as per the u-mass metric was again 198 and therefore, 198 features for each review were retrieved for the training set. The same model was then used to obtain features for validation and test set.

Table 1: Top 5 words and their allocated topics

Word	Probability	Topic
book	0.205	13
original	0.169	100
kid	0.158	167
woman	0.154	86
action	0.146	98

3. Machine Learning Models

In this section, a number of machine learning (ML) classifiers were trained and optimised in order to perform this binary classification. For the purpose of experimenting with various embeddings coming from different NLP tools, functions were created in which the models were trained, tuned and validated based on the given data inputs.

The followed possess are used for each model:

- Initialise and trained a baseline ml model
- Calculate the accuracy and F1 scores on the training and validation set
- Performed hyperparameter tuning using cross-validated grid-search over a specified parameter grid.

- Calculate the accuracy and F1 scores on the training and validation set on the optimised model

(Raza et al 2019)

The models that were chosen are the following:

3.1 Support Vector Machine

Support Vector Machine (SVM) is widely used for binary classification and outperforms other classifiers when the dataset contains class imbalance (Raza et al 2019). Also, the method is really robust against overfitting (especially for text data sets due to high-dimensional space (Kowsari et al 2019). For the purpose of the analysis, several hyperparameters have been fine-tuned such as the Kernel with its gamma and C parameters in order to adjust the misclassification penalty and the influence of the individual data point in the classification respectively (Scikit Learn 2022). Regarding the performance of the optimised model, the accuracy and F1 score seems to be higher with the topic to document embedding at 86.6% without overfitting the training set.

3.2 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a non-parametric technique widely used for text classification (Kowsari et al 2019). The classifier falls under the category of instance-based learning where training instances are used to make predictions (Raza et al 2019).For the fine-tuning of the classifier, a different number of neighbours were examined in order to adjust the smoothness of the decision boundary along with their weights parameters used by the classifier to make the prediction (Scikit Learn 2022). Regarding the performance of the optimised model the accuracy and F1 score seems to be higher with the Sklearn Vectorisation embedding at 85.1% however with significant overfitting to the training set.

3.3 Naive Bayes

Naive Bayes is a probabilistic classifier widely used for binary and multiclass text classification and it tends to perform better on datasets with fewer training points than other classifiers. Also, it is faster than the linear model and is suited for high-dimensional datasets(Naeem et al 2022) . For our analysis, we used the Gaussian and Multinomial variations of Naive Bayes due to the larger size of the dictionar (Raza et al 2019). Regarding the performances of the two models, the F1 score seems to be higher in the case of the Gaussian model with the Sklearn Vectorisation embedding at 87% without overfitting the training set.

3.4 Logistic Regression

Logistic regression is one of the most widely used parametric efficient probability-based text classifiers (Tsangaratos and Ilia 2016). For the hyperparameter tuning, different solvers based on their efficiency in working with large data were examined (Thangaraj and Sivakami 2018).Then different regularisation methods were applied such as L1 and L2 regularisations

along with the combination of two in order to reduce the complexity coming from a large number of training features (Hale 2019) (Nagpal 2017). Regarding the performance of the optimised model the accuracy and F1 score seems to be higher with the Sklearn Vectorisation embedding at 80.6% without overfitting the training set.

3.5 Ridge Regression

Ridge regression is a regularised version of standard linear regression and it is a popular solution for text classification (Raza et al 2019). It is also known to perform better than other classifiers when the dataset contains a small number of positive examples (imbalance) (Thangaraj and Sivakami 2018).For the optimisation of the model, the alpha parameter was fine-tuned which represents the strength of the model's regularisation. The F1 score of the optimised model reaches its highest levels with the Sklearn Vectorisation embedding at 87% without overfitting the training set (Scikit Learn 2022).

Table 2: Analysis of every Machine Learning algorithm

Classifier	Embedding	Accuracy Train	Accuracy Valid	F1 train	F1 Validation
KNN	Sklearn Vectorisation	1	0.750929	1	0.851770
SVM	Topic Modelling	0.767228	0.795799	0.867603	0.866667
Logistic Regression	Sklearn Vectorisation	0.676624	0.705390	0.786885	0.806825
Gaussian Naive bayes	Sklearn Vectorisation	0.763803	0.770446	0.866087	0.870341
Multinomial Naive Bayes	Topic Modelling	0.763715	0.763941	0.866030	0.866175
Ridge Classifier	Sklearn Vectorisation	0.782634	0.772305	0.875317	0.870439

4. Deep Learning Models

In order to approach Neural Networks for classification, 3 different kinds of networks were chosen to be trained with different forms of embeddings as inputs. Bidirectional LSTM , LSTM and Conv1D, and Dense.

Bidirectional LSTM - a Recurrent Neural Network Layer was chosen to understand if the user reviews may have nuances such as a word in a sentence affecting what the following word may be, and therefore, the overall classification of the review. The Bidirectional aspect would take into account the actual and reverse order of document-level embeddings and therefore, the word within (Liu and Guo 2019). The comparison between tensorflow

vectorization and topic to document probabilities is done to try to robustly choose the right embedding method for classification. The Tokenizer class of Keras is converted each review into integer sequence or a vector that has a coefficient for each token in the form of binary values (Sharma 2021). For the purposes of the analysis, `texts_to_sequences` on the document list was used. The text_to_sequences method helps in converting tokens of text corpus into a sequence of integers.

LSTM and Conv1D - a Recurrent Neural Network and Convolutional Neural Network layer to combine the learning of temporal aspects and then aggregating the high-level spatial features of a comment to classify as helpful or not helpful (Jacovi et al 2020).

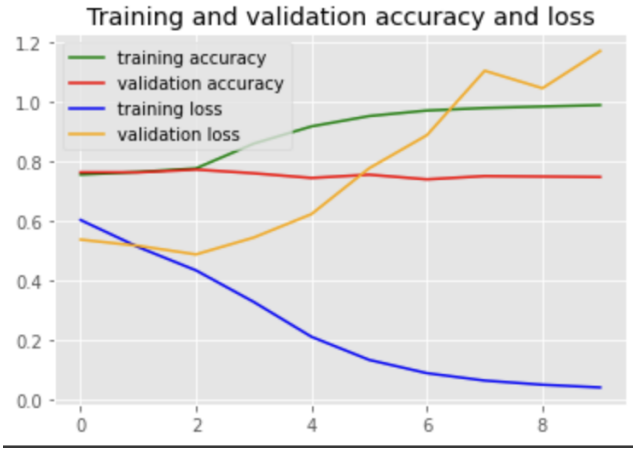
Lastly, Dense Neural Network builds on to pre-trained Bidirectional encoder representations from transformers (BERT) to holistically take into account context-level, word-level and sentence-level representation of a comment to adjust the final weights and biases and present the classification (Tensorflow 2022). To pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers (Sun et al 2019).

The experiments were done using 3 different embedding structures - Tensorflow Vectorization, Topic to Document probabilities, and BERT (uncased). The final Neural Networks were: Bidirectional LSTM (Tensorflow Vectorization), Bidirectional LSTM (Topic to Document probabilities), LSTM and Conv1D (Tensorflow Vectorization), LSTM and Conv1D (Topic to Document Probabilities), and Dense Neural Network (BERT).

4.1 Keras Tokenizer

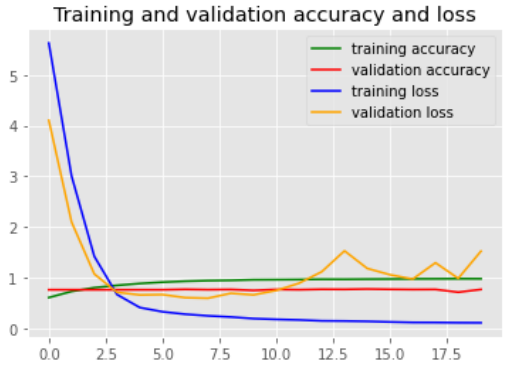
4.1.1. Bidirectional LSTM

The model was trained on the train data set for 10 epochs. Not long after the first epochs, the model showed a significant overfitting, reaching by the end of its train, a train accuracy of ~98% while the validation accuracy was measuring up to ~74%.



4.1.2. LSTM-Conv1D

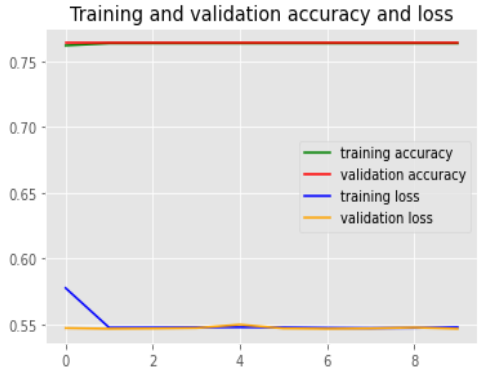
Dropouts, regularisations and batch normalisations were implemented while using extra 1D convolutional layers. Therefore, the model was still trained for 10 epochs providing the same results as a simple LSTM. The model showed significant overfitting, measuring a training accuracy of ~98% and validation accuracy at 77% the 20th epoch.



4.2 Topic to Document Probabilities

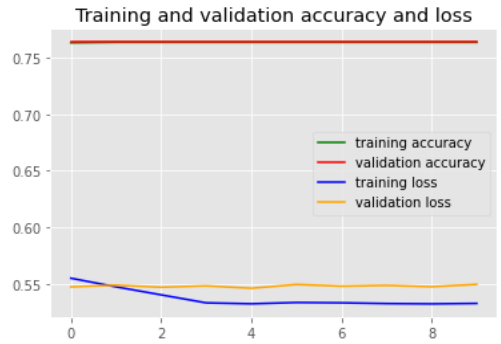
4.2.1. Bidirectional LSTM

The model showed very minimal learning through the epochs, as the accuracies remained quite flat (Train: 0.7637 Valid: 0.7639) and so did the losses. Although, the model did not overfit as can be seen in the accuracies and also had a good F1 score of 0.86617.



4.2.2. LSTM-Conv1D

The results were exactly the same as in Bidirectional LSTM. But here, the losses were quite far apart in the end, potentially indicating overfit.



4.3 BERT (uncased)

Unoptimised:
The pre – trained BERT model was implemented on the raw data (train.csv). The model was trained for 5 epochs and a batch size of 32. Compared to the Keras Tokenizer models, this model showed no overfitting, and measured an F1 score of 0.8661. However, the model showed no learning progress leading the analysis to further extension by fine – tuning it.

Optimised:
The optimisation was done by changing the optimiser to ADAMW on BERT model to integrate the weight decay and learning rate (Tensorflow 2022). The model was trained for 10 epochs. By the end of the final epoch, it measured a training accuracy of 0.7637 and a validation accuracy of 0.7639. An F1 score of 0.86 was recorded. Although the accuracies were flat, the losses reduced gradually and altogether, but still the losses are slightly apart.

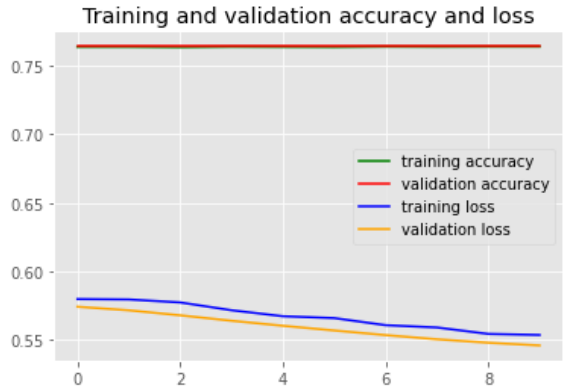


Table 3: Analysis of the Deep Learning models

Deep Learning Classifiers	Embedding	Accuracy Train	Accuracy Valid	F1 Validation
Bidirectional LSTM	Keras Tokenizer	0.9836	0.7477	0.8666
LSTM - Conv1D	Keras Tokenizer	0.9836	0.7778	0.8666
Bidirectional LSTM	Topic Modelling	0.7637	0.795799	0.866667
LSTM - Conv1D	Topic Modelling	0.7637	0.7639	0.86617
BERT unoptimised	Raw Data	0.7637	0.7639	0.86617
BERT optimised	Raw Data	0.7637	0.7639	0.8661

5. Final Model: Multinomial Naive Bayes with CountVectorization

Based on the final Training and Validation accuracies and F1 scores, the Naive Bayes has been chosen for the following reasons: The Deep Learning models showed strange training, despite the different structures and inputs of the networks - the metrics have been very similar and training very flat. The IMDB dataset on hand is quite imbalanced and has not got as many observations to use well for a Neural Network. Naive Bayes is suitable for smaller datasets and for high-dimensional datasets (as in the case with the CountVectorization approach) (Raza et al 2019). Whilst other machine learning models were trained as well and showed similar performance, the generative model was computationally efficient whilst preserving the quality of metrics, moreover, it is a scalable model. There was also more reliability in the prediction because the classification split on the testing data was also close to that of the training data (73% and 76% respectively).

Hence, the final model:

Multinomial Naive Bayes (CountVectorization) with Validation F1 = 0.866175

References

1. ACM Transactions on Intelligent Systems and Technology. 2022. A Survey on Text Classification: From Traditional to Deep Learning | ACM Transactions on Intelligent Systems and Technology. [online] Available at: <https://dl.acm.org/doi/10.1145/3495162> [Accessed 20 July 2022].
2. Analytics Vidhya. 2022. Text Classification using BERT and TensorFlow. [online] Available at: <https://www.analyticsvidhya.com/blog/2021/12/text-classification-using-bert-and-tensorflow/#:~:text=BERT%20is%20a%20very%20good,%2Dof%2Dthe%2Dart.> [Accessed 20 July 2022].
3. Brownlee, J., 2022. A Gentle Introduction to the Bag-of-Words Model. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/gentle-introduction-bag-words-model/> [Accessed 19 July 2022].
4. Devlin, J., Chang, M., Lee, K. and Toutanova, K., 2022. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online] arXiv.org. Available at: <https://arxiv.org/abs/1810.04805> [Accessed 20 July 2022].
5. Hale, 2019. Don't Sweat the Solver Stuff. [online] Medium. Available at: <https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451> [Accessed 18 July 2022].
6. Ieeexplore.ieee.org. 2022. Deeplearning Model Used in Text Classification. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/8632592> [Accessed 20 July 2022].
7. Jacovi, A., Shalom, O. and Goldberg, Y., 2020. Understanding Convolutional Neural Networks for Text Classification. [online] Available at: <https://arxiv.org/pdf/1809.08037.pdf> [Accessed 16 July 2022].
8. Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes and Brown, 2019. Text Classification Algorithms: A Survey. Information, 10(4), p.150.
9. Li, Z., Shang, W. and Yan, M., 2016. News text classification model based on topic model. 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS),.
10. Liu, G. and Guo, J., 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. Neurocomputing, 337, pp.325-338.
11. Medium. 2022. A Simple Explanation of the Bag-of-Words Model. [online] Available at: <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971> [Accessed 19 July 2022].
12. Medium. 2022. A Simple Explanation of the Bag-of-Words Model. [online] Available at: <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971> [Accessed 20 July 2022].
13. Medium. 2022. An introduction to Bag of Words and how to code it in Python for NLP. [online] Available at: <https://medium.com/free-code-camp/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04> [Accessed 19 July 2022]
14. Medium. 2022. BERT Explained: A Complete Guide with Theory and Tutorial. [online] Available at: <https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c> [Accessed 20 July 2022].
15. Medium. 2022. Classification using Pre-trained Bert Model (Transfer Learning). [online] Available at: <https://medium.com/@yashvardhanvs/classification-using-pre-trained-bert-model-transfer-learning-2d50f404ed4c> [Accessed 20 July 2022].

16. Medium. 2022. Fine-Tuning BERT for Text Classification. [online] Available at: <https://towardsdatascience.com/fine-tuning-bert-for-text-classification-54e7df642894> [Accessed 20 July 2022].
17. Medium. 2022. How to Do Text Binary Classification with BERT?. [online] Available at: <https://towardsdatascience.com/how-to-do-text-binary-classification-with-bert-f1348a25d905> [Accessed 20 July 2022].
18. Medium. 2022. Multi-Class Text Classification Model Comparison and Selection. [online] Available at: <https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568> [Accessed 20 July 2022].
19. Medium. 2022. Sentiment Analysis using LSTM and GloVe Embeddings. [online] Available at: <https://towardsdatascience.com/sentiment-analysis-using-lstm-and-glove-embeddings-99223a87fe8e> [Accessed 20 July 2022].
20. Medium. 2022. Step by step building a multi-class text classification model with Keras. [online] Available at: <https://medium.com/swlh/step-by-step-building-a-multi-class-text-classification-model-with-keras-f78a0209a61a> [Accessed 20 July 2022].
21. Medium. 2022. Text Classification in Keras (Part 1) — A Simple Reuters News Classifier. [online] Available at: <https://towardsdatascience.com/text-classification-in-keras-part-1-a-simple-reuters-news-classifier-9558d34d01d3> [Accessed 20 July 2022].
22. Medium. 2022. Text classification made easy with AutoKeras. [online] Available at: <https://towardsdatascience.com/text-classification-made-easy-with-autokeras-c1020ff60b17> [Accessed 20 July 2022].
23. Naeem, M., Rustam, F., Mehmood, A., Mui-zzud-din, Ashraf, I. and Choi, G., 2022. Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms. PeerJ Computer Science, 8, p.e914.
24. Nagpal, 2017. L1 and L2 Regularization Methods. [online] Medium. Available at: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c> [Accessed 20 July 2022].
25. Paperswithcode.com. 2022. Papers with Code - On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. [online] Available at: <https://paperswithcode.com/paper/on-the-role-of-text-preprocessing-in-neural> [Accessed 20 July 2022].
26. Raza, M., Hussain, F., Hussain, O., Zhao, M. and Rehman, Z., 2019. A comparative analysis of machine learning models for quality pillar assessment of SaaS services by multi-class text classification of users' reviews. Future Generation Computer Systems, 101, pp.341-371.
27. Schofield, A., Magnusson, M. and Mimno, D., 2017. Pulling Out the Stops: Rethinking Stopword Removal for Topic Models. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers,.
28. Scikit Learn, 2022. 1.4. Support Vector Machines. [online] scikit-learn. Available at: <https://scikit-learn.org/stable/modules/svm.html#svm-kernels> [Accessed 14 July 2022].
29. Scikit Learn, 2022. sklearn.neighbors.KNeighborsClassifier. [online] scikit-learn. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> [Accessed 16 July 2022].
30. Sharma, P., 2021. [online] Machinelearningknowledge.ai. Available at: <https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit_on_texts-texts_to_sequences-texts_to_matrix-sequences_to_matrix/> [Accessed 15 July 2022].

31. Sun, C., Qiu, X., Xu, Y., Huang, X. (2019). How to Fine-Tune BERT for Text Classification? In: Sun, M., Huang, X., Ji, H., Liu, Z., Liu, Y. (eds) Chinese Computational Linguistics. CCL 2019. Lecture Notes in Computer Science (), vol 11856. Springer, Cham. https://doi.org/10.1007/978-3-030-32381-3_16
32. Tensorflow, 2022. Classify text with BERT | Text | TensorFlow. [online] TensorFlow. Available at: <https://www.tensorflow.org/text/tutorials/classify_text_with_bert> [Accessed 19 July 2022].
33. TensorFlow. 2022. Basic text classification | TensorFlow Core. [online] Available at: <https://www.tensorflow.org/tutorials/keras/text_classification> [Accessed 20 July 2022].
34. TensorFlow. 2022. tf.keras.layers.LSTM | TensorFlow Core v2.9.1. [online] Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM> [Accessed 20 July 2022].
35. Thangaraj, M. and Sivakami, M., 2018. Text Classification Techniques: A Literature Review. Interdisciplinary Journal of Information, Knowledge, and Management, 13, pp.117-135.
36. Tsangaratos, P. and Ilia, I., 2016. Comparison of a logistic regression and Naïve Bayes classifier in landslide susceptibility assessments: The influence of models complexity and training dataset size. CATENA, 145, pp.164-179.