Classifying Dialogue to Characters and EFR in TV Shows

arnav21019@iiitd.ac.in

May 13, 2024

1 Problem Statement

This project aims to develop a natural language processing system capable of analyzing dialogues in TV show transcripts to identify the speakers, and classify their utterances, leveraging datasets comprising character names and their corresponding lines of dialogue, the system will employ techniques such as named Tokenisation, Relevant Topics from Sentiment Analysis, and Term Frequency - Inverse Document Frequency to attribute each line to its respective speaker and infer the emotional state conveyed in the same.

Now, I understand that it's not theoretically possible to correctly classify each line to its speaker with 100 percent accuracy, especially since shorter lines, grunts, one word responses etc. are liable to be spoken by any character, there would be no way to uniquely classify them to a single character, and by definition, the more dominant class is liable to usurp the sentence as its own. However, What I posit, is that each character has a unique way of communicating what's on their mind, the way they choose their words reflects this, and in this project, I'll try to confirm whether or not this hypothesis is unreasonable.

I understand that this project doesn't seem to have any real-life applications, but then again, most of the project ideas for this course aren't particularly novel. While the project may not have direct practical applications, it helps provide insights into the nuances of language usage and character development in storytelling, which could be valuable for writers, researchers, or even fans who want to dive deeper into the shows they love.

2 Literature Review

Sentiment Analysis: This project is rather similar to Sentiment Analysis, since we're trying to classify different texts to classes, bases on their labels. To get a broad understanding of the subject, I went through A survey on sentiment analysis methods, applications, and challenges by Wankhade et al., which covered important approaches and tasks involved in SA, including Lexicon and ML based approaches. I also briefly went through Sentiment Analysis in the News by Balahur, Steinberger et al., especially since their dataset shared multiple aspects with this project's dataset, to understand multi-sized word windows, for classification.

Text Classification: I covered 'Text Classification Using Machine Learning Techniques' by Ikonomakis et al., since the diagram they borrowed from Sebastiani, was primarily responsible for informing my understanding of the text classification, Read Document - Tokenise Text - Stemming - Deleting Stop Words - Feature Selection - Apply ML Algorithm

Introduction to NLP and Deep Learning Models: I primarily covered 'The Sequential model' on the Keras website and had a cursory reading of 'A comparative review on deep learning models for text classification' by Ghazali et al. since I tried to cover the TF Keras Sequential Model as one of the models for classification, I also read the huggingface docs for the uncased version of BERT, and all the associated documentation, but was unable to train it on my machine owing to computing restraints.

3 Methodology

To tackle the this problem, I utillised 6 different machine learning models: Multinomial Naive Bayes, Random Forest, Logistic Regression, Multi-Layer Perceptron, a Voting Classifier ensemble, and a Sequential ANN model. I used multiple models so as to compare their performance and evaluate their strengths and weaknesses in handling this specific task.

3.1 Model 1: Multinomial Naive Bayes

Multinomial Bayes Classification assumes that the features, in text classification, follow a multinomial distribution, accommodating multiple outcomes instead of just two, calculates the probability of text belonging to a class based on the frequency of words, further assuming that these word counts are independent given the class, but even despite the highly probable violations of this independence assumption, it is often effective in practical applications.

3.2 Model 2 : Logistic Regression

Logistic regression, when used for multi-class classification, predicts the probability of an instance belonging to each class within a set of multiple classes, extending binary logistic regression by employing multiple logistic functions, each representing the probability of an instance belonging to a specific class versus all other classes combined. The model then assigns the class with the highest probability as the predicted class for each instance.

3.3 Model 3 : Random Forest

Random Forest is an ensemble method which, utillises multiple deep decision trees, and trains different portions of the complete datsets on each tree, it consequently averages them, which ends up reducing the standard deviation considerably, albeit, along with an uptick in the model's bias.

3.4 Model 4 : Simple Multi-Layer Perceptron

The Simple Multi-Layer Perceptron optimizes the log-loss function using the 'adam' solver, a stochastic gradient-based optimizer proposed by Kingma and Ba. During training, the solver iteratively updates model parameters by computing the partial derivatives of the loss function with respect to these parameters at each time step, aiming to minimize the loss and improve model performance.

3.5 Model 5 : Ensemble Voting: MNB+LR+RF+MLP

This model simply implements a Voting Classifier, which takes into account the results from the preceding models, and further implements a hard-voting scheme, which was chosen since the results for the preceding models are largely similar, when solely considering the top-3 classes, over 200 iterations.

3.6 Model 6 : Sequential ANN

The Keras Sequential stacks a bunch of layers, allowing for the hard-coded layer-wise construction for neural networks, adding them sequentially, from input to output, with each layer connected to the previous one, using a plethora of pre-defined layers, activation functions, and optimizers for classification and regression.

4 Dataset

I initially used the 'Seinfeld Chronicles' dataset, which was last updated on Kaggle, by user Aman Srivastava, around 6 years back. It was extracted from a fan website dedicated to the American Sitcom 'Seinfeld.' Later, I decided to incorporate another dataset, ' The Office (US) - Complete Dialogue/Transcript' by Nasir Khalid on Kaggle since the data is comparatively, 'more balanced', and features a greater variety of 'main' characters, which reduces dialogue similarity. For each dataset, I have more than a 100,000 samples to train on, for the top 4 and 5 characters, respectively (on the

basis of who's spoken most), I have also performed downsampling and upsampling on the characters to ensure informative gradients, and prevent over-dominance of a single character, Tokenization and TF-IDF were essential components for the text preprocessing, Tokenization helped parse the text into individual tokens, while TF-IDF assigned weights to each token based on its frequency across documents, highlighting their significance while mitigating the influence of common words. This approach particularly helped in feature extraction, enabling the derivation of meaningful features from text for accurate modeling and classification tasks, Though I initially used the top-10 characters for each dataset for classification, neither the neural networks, oversampling, nor undersampling could help in creating better results for characters at the bottom end. This led me to eventually only use the top-3 (and later top-4) characters for classification.

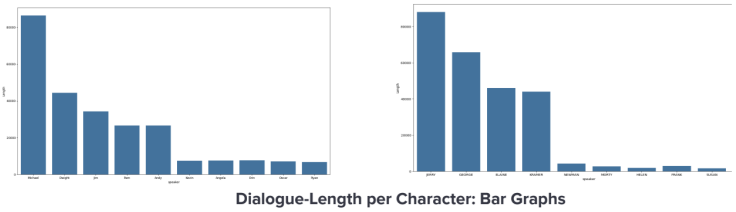

Figure 1: Characters, and the number of lines they speak

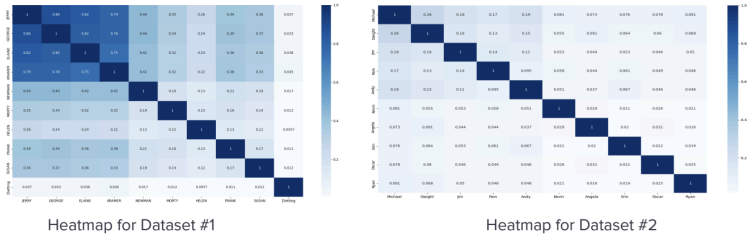

Figure 2: Textual correlation b/w different characters' lines

5 Results and Observations

5.1 Results

Accuracy scores for the aforementioned models are shown in Table 1. The results indicate an overall similar performance insofar as random forest, logistic regression, multinomial naive Bayes, and multi-layer perceptron are concerned. There is a small uptick when the ensemble model is used, but the greatest increment is observed when switching to the sequential ANN, suggesting it is better suited for classifying textual data.

Model	Seinfeld	The Office
Random Forest	0.472	0.45
Multinomial Naive Bayes	0.42	0.49
Logistic Regression	0.43	0.49
Multi Layer Perceptron	0.51	0.49
Ensemble Voting Classifier	0.51	0.49
Sequential ANN	0.7239	0.7435

Table 1: Accuracy Scores

Figure 3: The predictor correctly classifying a dialogue to Jim in TheOffice.csv

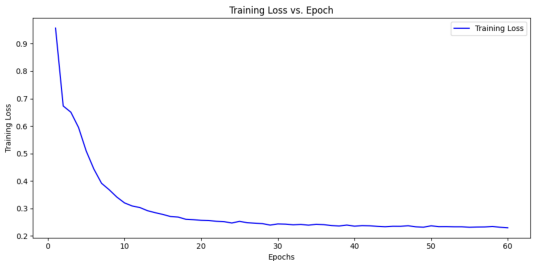

Figure 4: ANN Training Loss vs Epoch plot: Dataset 1

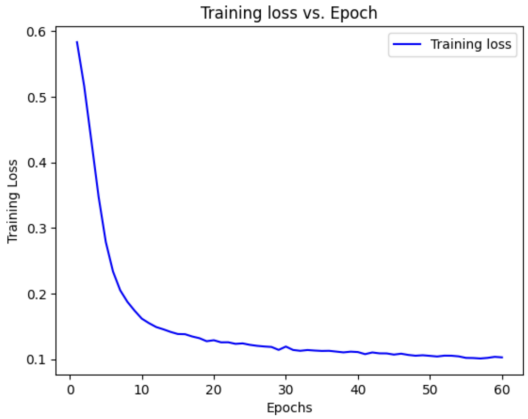

Figure 5: ANN Training Loss vs Epoch plot: Dataset 2

5.2 Observations

In the "traditional" models, The Second Dataset performed slightly better than the first, this can likely be attributed to the ginormous difference b/w the Dialogue Length for Jerry and the rest of the main cast in the Seinfeld Chronicles dataset. Furthermore, The similarity b/w the dialogues in the first dataset are much more pronounced. However, These differences don't lead to a gigantic difference in the former's performance using the Keras Sequential NN, which goes on to affirm the model's superiority at classifying imbalanced data, without the need for under or over sampling. Furthermore, Most sentences of considerable size, that I inputted in the predictor function for the "traditional" models, gave surprisingly accurate results, which is counterintuitive, since individual probabilities decline with each additional token. However, It just might be responsible for distinguishing b/w characters, as the empirical results would suggest. In most class exercises, the overall accuracy required to deem a solution fit is somewhere north of 50-51 percent, and since the ensemble model, and the sequential neural network beat that threshold, I would like to think, that my hypothesis, is somewhat reasonable, at differentiating between classes/characters, especially when newwer and more complex ML models are utillised.

6 Future Work

While the methodology demonstrates promising results, there are avenues for further improvement and exploration. Enhancing the models' accuracy and robustness could involve refining the feature engineering process and exploring alternative architectures for multi-label classification . Additionally, addressing inherent limitations, such as potential biases in training data and the risk of over-reliance

on shorter dialogues, remains crucial. Future research could focus on diversifying training datasets and implementing more sophisticated evaluation strategies to mitigate these challenges. Exploring various Transformers and Neural Networks to find the most suitable for text classification can optimize model performance, aligning with the project's objectives.

References

Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Zulqarnain, M., Ghazali, R., Hassim, Y. M. M., Rehan, M. (2020). A comparative review on deep learning models for text classification. Indones. J. Electr. Eng. Comput. Sci, 19(1), 325-335.

Ikonomakis, M., Kotsiantis, S., Tampakas, V. (2005). Text classification using machine learning techniques. WSEAS transactions on computers, 4(8), 966-974.

Wankhade, M., Rao, A. C. S., Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. Artificial Intelligence Review, 55(7), 5731-5780.

Balahur, A., Steinberger, R., Kabadjov, M., Zavarella, V., Van Der Goot, E., Halkia, M., ... Belyaeva, J. (2013). Sentiment analysis in the news. arXiv preprint arXiv:1309.6202.