CS178 Project Group 7 Final Report
IMDb Review Analysis and Data Mining with Bert Mode

1.1 Introduction

Text classification is a subfield of machine learning. Computers are trained on text data to classify them into different categories by recognizing the patterns and determining the context of each input. One application of the text classification method is on the IMDb Review Dataset using the Bert Model. Bert Model will be used to classify textual review into either negative or positive classes.

1.2 Task Distribution

Ajwad: Bert Model
Ivonne: Data Understanding
Safia: Data Understanding and Bert Model Cluster

2.1 Data Understanding

The IMDb dataset is a collection of 50,000 reviews made by real users. Of these, 25,000 are classified as Negative, and the other 25,000 are classified as Positive. For data understanding, only half of the raw dataset is being used, with 12,500 positive reviews and 12,500 negative reviews.

Initialization

Half of the raw IMDb review data is processed and placed into a data frame in the program containing the review textual content and the label of each review. Negative reviews are labeled as class 0 while Positive reviews as class 1.

	review	label
	Bromwell High is a cartoon comedy. It ran at t...	1
	Homelessness (or Houselessness as George Carli...	1
	Some films that you pick up for a pound turn o...	0
	This is one of the dumbest films, I've ever se...	0

Fig 1 Data Frame Visualization

Next, the number of characters in each review is measured and recorded to create a visual representation of the length distribution for both Negative and Positive reviews. On average both Positive and Negative reviews are around 1,300 characters. The longest positive reviews are 13,704 characters, while the longest negative reviews are 8,969 characters long. The shortest positive reviews are 70 characters and negative reviews are 52 characters.

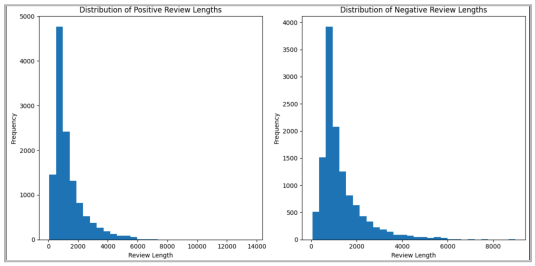

Fig 2 Distribution of positive and negative review length

Word Cleaning

Word Cleaning refers to transforming raw textual data into processable. This process includes removing non-alphabet elements, lower-casing all alphabets, removing stopwords, and lemmatizing each word. Lemmantizing refers to the process of transforming words into their base form, also known as token. The tokens in each review will be combined and stored in the data frame.

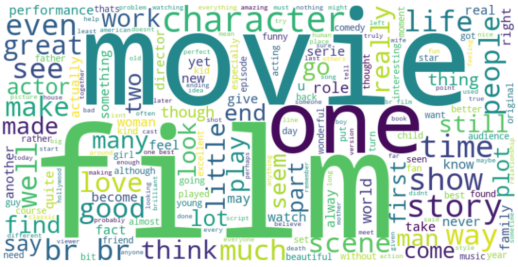

Fig 3 Word Cloud for Positive Reviews

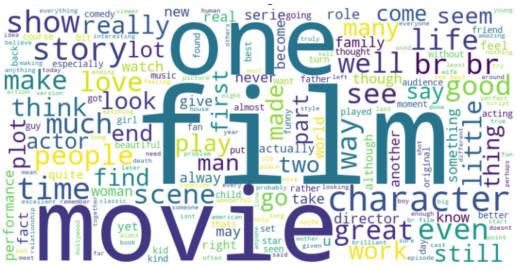

Fig 4 Word cloud for Negative Reviews

The token composition in Positive and Negative reviews is similar. Thus, further data processing is required to create a unique token composition for each class.

Scoring

The cosine similarity of the TF-IDF score of each token in each class is used to determine whether there is a significant difference in token composition between the two classes. TF-IDF score refers to the product of the TF values and IDF values. TF and IDF values are calculated using the following equations:

$$\text{TF} = \frac{\text{token frequency in class}}{\text{total token in class}}$$

$$\text{IDF} = \log\left(\frac{1 + \textit{total documents}}{1 + \textit{documents with token}}\right)$$

The TF-IDF of each token in positive class and negative class will each form a vector. Cosine Similarity will measure the angle between the two vectors whereas the closer the Cosine Similarity value to 1, the more similar the two classes are. The Cosine Similarity can be calculated using the following equation:

$$\frac{\text{Positive TF-IDF} \cdot \text{Negative TF-IDF}}{\|\text{Positive TF-IDF}\| \, \|\text{Negative TF-IDF}\|}$$

The method that yields the most unique set of words is the method with the lowest Cosine Similarity score.

2.1.1 10 Common Words Removal

Based on the frequency of each token in both classes, 10 tokens with the highest frequencies are removed.

Fig 5 Word Cloud for Positive Reviews

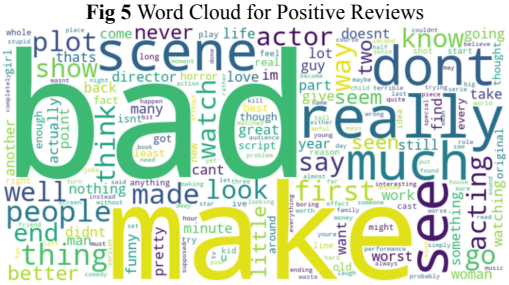

Fig 6 Word Cloud for Negative Reviews

2.1.2 Manhattan Distance

Manhattan Distance is the similarity between two values by calculating how similar the token frequency between the two classes using the equation below:

$$\frac{\text{Negative Frequency} - \text{Positive Frequency}}{max(\text{Negative Frequency}\ ,\ \text{Positive Frequency})}$$

Fig 7 Word Cloud for Positive Reviews

Fig 8 Word Cloud for Negative Reviews

2.1.3 TF-IDF and Cosine Similarity

For each token in the data frame, the program will compare the cosine similarity values for two data sets where it either includes or does not include the token. If the data set without the token has a lower cosine similarity, then the token will be removed. Otherwise, the token will be left on the dataset. This process will stop when there is no significant difference between the mean of the previous 1000 values and the current similarities value.

Fig 9 Word Cloud for Positive Reviews

Fig 10 Word Cloud for Negative Reviews

2.1.4 Result and Conclusion

Method	Score
Original	0.93
10 Common Words Removal	0.93
Manhattan Distance	0.53
Cosine Similarity Reduction	0.75

Based on the score above, the Manhattan Distance method yields the lowest similarity score compared to the other methods. It shows that Manhattan Distance performs better than other methods in extracting unique tokens for each class.

2.2 Bert Model

The Bidirectional Encoder Representations from Transformers, or BERT Model is an open-source Machine Learning framework designed by Google to invent a new way to perform Natural Language Processor tasks. BERT Model studies the context of a text from left to right and right to left simultaneously, allowing BERT to understand textual context more effectively. The model is pre-trained on a massive corpus, which increases its performance. The BERT model is used to analyse IMDb Review dataset due to its ability to perform excellently in understanding textual context. The model does not require manual feature extraction due to its ability to generate embeddings with complex relationships in the text. In this project, bert-based-uncased, a pre-trained BERT model, is used to take advantage of its extensive training on the text

corpora, a set of text data that are used to train the model.

2.2.1 Training

The model is trained using two types of split: 50-50 split and 60-20-20 split. The 50-50 split is further split into 40-10-50 split. 40% of the data will be the training data, 10% will be the validation data and the remaining 50% will be the test data. The model is trained on 40% of the data twice: the first time is trained on the raw data and the second time on cleaned data. Additionally, each run's performance will be measured after three epochs and five epochs to see whether cleaning data and increasing epochs will increase the model's performance. The same runs are done on a 60-20-20 split, where 60% are used for training, 20% are used for validation, and the remaining 20% for testing. The performance between the two splits will be compared for each run to see if there is a correlation between data splitting and performance.

2.2.2 Parameters

The BERT model is trained using parameters suggested in the original paper, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." For smaller datasets, training for 3 to 5 epochs is suitable when using a pre-trained model like BERT, and we choose epochs of size 3 and 5 to evaluate their performance and see if there are significant differences between the two methods. AdamW optimizer with a learning rate of 2e-5 and 5e-5 and batch sizes of 16 was used as the original paper suggested. Additionally, a linear learning rate scheduler from Hugging Face is used to decrease the learning rate after each batch.

2.2.3 Performance

For the first run on a 40-10-50 data split and 3 epochs, the model yielded a 93.97% accuracy.

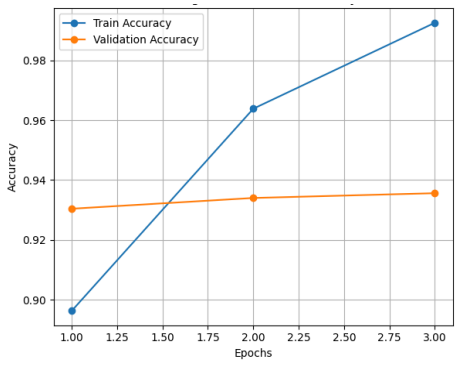

Fig 11 Training and Validation Accuracy Graph for 3 epochs

The model's learning rate was changed from 5e-5 to 2e-5 to see if this method would improve the accuracy. The model yielded a 93.94% accuracy with no significant difference, so the learning rate was reverted to 5e-5.

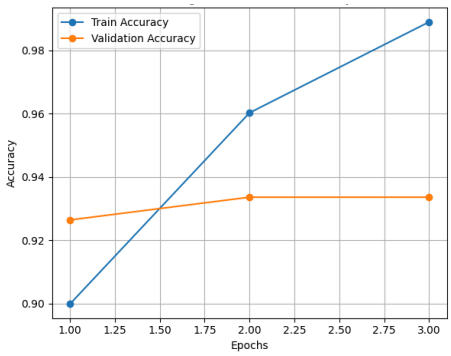

Fig 12 Training and Validation Accuracy Graph for smaller learning rate

For the next run, the model was trained with 5 epochs. The test accuracy yielded was 93.58% and had no significant improvements.

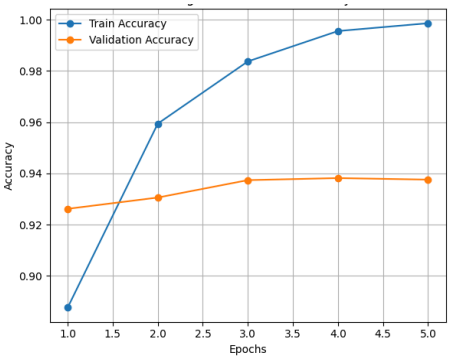

Fig 13 Training and Validation Accuracy for 5 epochs

Some HTML tags appeared on the word clouds from the first two runs, which might have affected the model's accuracy. Thus, the training data will be cleaned to improve the model's accuracy. The cleaning process involves lower-casing all words and removing special or repetitive characters such as HTML tags and stopwords. Running on 3 epochs, the cleaned data set results in an accuracy of 93.60%.

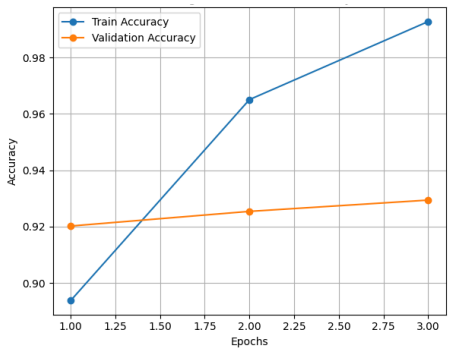

Fig 14 Training and Validation Accuracy for cleaned data set

The last run was done with a 60-20-20 data split to find if there is any correlation between training data size and performance. The result yields an accuracy of 93.83%.

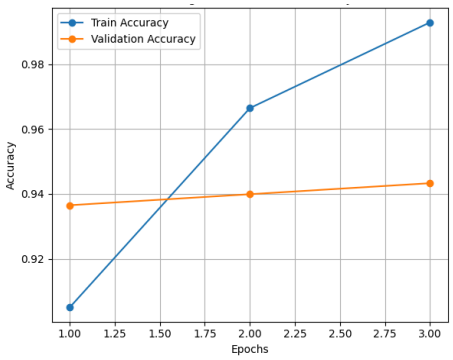

Fig 15 Training and Data Validation for 60-20-20 data split

In conclusion, there is no significant correlation between the methods above and performance which results in a significant increase in accuracy. This might be due to the nature of BERT's pre-trained model which was trained on 3.3. billion words such that training and an additional 50k reviews will not create any significant impact.

2.3 Bert Embeddings K-Clustering Model

BERT embeddings can capture semantic information from the IMDb review data set. These embeddings represent the text as a numerical vector in a high-dimensional space. It makes the data suitable for clustering. Thus, the K-means clustering was used to group similar TV shows based on their semantic meanings. Additionally, t-SNE (t-distributed Stochastic Neighbor Embedding) reduced the dimensionality of the embedding for visualization purposes. K-means clustering effectively groups reviews based on shows using embedding similarity. The Elbow method was also used to determine the optimal number of clusters for graphs.

2.3.1 Training

For the training process, the embeddings were generated using a pre-trained BERT model previously trained on the original raw data of the IMDB review dataset with a 30-20-50 split. Since K-means is an unsupervised training model, it does not require training in the traditional sense but rather fits the data to identify cluster centroids. Using the pre-trained BERT model, the model can capture semantic representations without additional finetuning.

2.3.2 Parameters

The number of clusters was determined using the elbow method. The optimal value was 2 since the plot of inertia showed the steepest drop between clusters 1 and 2. We used t-SNE to reduce the dimensionality of the embeddings for 2D visualization. While t-SNE is computationally expensive, it provides a clear representation of clustering for analysis.

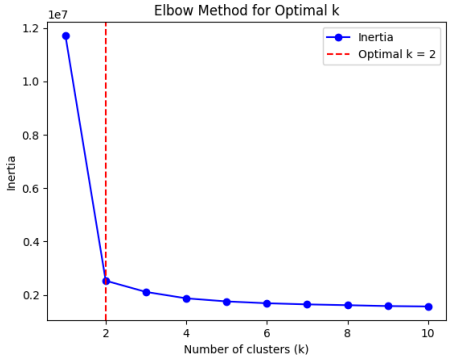

Fig 16 Elbow Method for Optimal k value

2.3.3 Performance

The data was divided into two primary clusters. In cluster 0, the majority (23799) of the labels were 0, with only 754 labeled as 1. Cluster 1 primarily consisted of labels 1 (24246), with 1201 of the labels being 0.

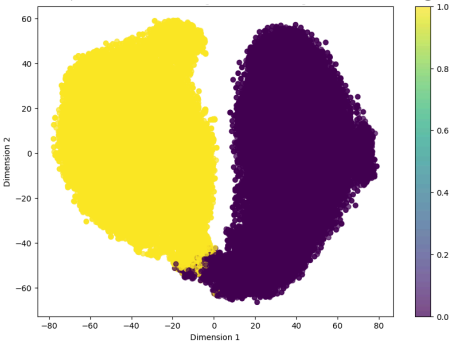

Fig 17 K-mean clustering of embeddings

References

[1] Devlin, Jacob. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[2] Geelen, Pieter. “Knee-Plot Algorithms: Standardizing the Trade-off Dilemma.” *Medium*, **AI monks.io**, 31 July 2023, medium.com/aimonks/knee-plot-algorithms-standardizing-the-trade-off-dilemma-72f53afd6452.

[3] Subakti, A., Murfi, H. & Hariadi, N. The performance of BERT as data representation of text clustering. J Big Data 9, 15 (2022). https://doi.org/10.1186/s40537-022-00564-9