Spoiler Detection System Using BERT

Atharva Joshi
Harshitha Nooli

***Abstract*—In the age of digital content consumption, online reviews play a crucial role in influencing the decisions of movie enthusiasts. However, the proliferation of movie reviews on various platforms and social media websites has led to an increase in the volume of spoiler-filled content. Spoilers, which reveal critical plot information about a movie or show, can significantly diminish the viewer's enjoyment of the content. We aim to address this issue by developing a robust spoiler detection system that can automatically identify potential spoilers within reviews. We have employed machine learning models such as Decision Trees, Multinomial Naive Bayes classifier and custom built BERT (Bidirectional Encoder Representations from Transformers) models. By achieving this, we hope to enhance the experience of reading online movie reviews while safeguarding the narrative surprises for viewers and explore the best possible approach for solving the spoiler detection problem.**

Keywords : Spoiler detection, Decision Trees, Multinomial Naive Bayes Classifier, Natural Language Processing, Bidirectional Encoder Representations from Transformers (BERT)

I. Introduction

The exponential growth of online platforms and social media has revolutionized the way audiences engage with movies. Amidst this digital discourse, movie reviews play a pivotal role in shaping viewers' decisions and perceptions. However, the pervasive threat of spoilers has emerged as a critical concern, impacting the very essence of the cinematic experience. Spoilers, inadvertent revelations of key plot details, have the power to influence audience expectations, often diminishing the surprise and suspense carefully crafted by filmmakers.

In response to this challenge, our approach delves into the development of a sophisticated Spoiler Detection System. It addresses the imperative need to safeguard the movie-watching experience by leveraging advanced machine learning techniques. This involves detecting spoilers in a review using various machine learning models such as decision tree analysis, multinomial naive bayes classifier and custom built BERT natural language processing model by focusing on the identification and mitigation of potential spoilers within user-generated movie reviews. We have conducted our methodologies on the IMDB spoiler dataset.

We have experimented with decision trees, multinomial naive bayes classification and with the natural language processing model BERT which is employed by building a custom neural network architecture on top of the BERT based features. The existing work includes spoiler detection analysis using supervised classifiers in particular the support vector machines (Boyd-Graber et al., 2013). (Wan et al., 2019) have presented a paper on fine grained spoiler detection from large scale review corpora, where they have focused on SVM, CNN, SVM-BOW and HAN as the baseline models. (Lindo, 2020) has written an article on movie spoiler classification over online commentary, where support vector machines, logistic regression, naive bayes, random forest and Bi-LSTM model with pre trained GloVe embedding model have been proposed. (Wang et al., 2023) have proposed movie review spoiler detection using a multi view spoiler detection framework (MVSD). They built three subgraphs: movie-review subgraph, user-review subgraph, and knowledge subgraph, each modeling one aspect of the spoiler detection process. They have separately encoded the multi-view features of these sub-graphs through heterogeneous GNNs, then fused the learned representations of the three subgraphs through subgraph interaction. (Jeon et al., 2016) have worked on SVM based prediction model for determining spoilers presence in TV program tweets. The use of natural language processing model BERT has been very rarely used in spoiler detection and we have used a unique combination of neural network architecture on top of the BERT model to detect spoilers efficiently based on not only words but also the context behind the words.

As we navigate this exploration, we are propelled by a dual commitment – to enhance the quality of user interactions on online movie platforms and to contribute to the broader landscape of natural language processing models through Bidirectional Encoder Representations from Transformers. Through the lens of spoiler detection, we aim to redefine the boundaries of movie review analysis, presenting a paradigm shift that aligns with the evolving dynamics of digital communication. By amalgamating these diverse methodologies, we aspire to not only detect spoilers accurately but also to provide users with a nuanced understanding of the review context.

This paper unfolds the intricacies of our Spoiler Detection System, using decision tree analysis, multinomial naive bayes classification, and state-of-the-art BERT (Bidirectional Encoder Representations from Transformers) natural language processing. As we embark on this journey, our objectives encompass not only the technical prowess of our system but also its real-world impact. We seek to elevate the user experience, foster a spoiler-aware online environment, and contribute valuable insights to the academic discourse surrounding machine learning, contextual understanding in solving a problem and natural language processing.

II. Methodology

In this section, we delineate the methodology employed for the development of the Spoiler Detection System. The approach integrates multiple techniques, including TF-IDF vectorization fordecision tree analysis, multinomial naive Bayes, and BERT Tokenizer for a neural network architecture built on the BERT model. Each method plays a distinct role in enhancing the system's ability to identify and flag potential spoilers within movie reviews.

A. Term Frequency-Inverse Document Frequency Vectorization

TF-IDF is a numerical statistic that reflects the importance of a term in a document relative to a collection of documents. It is widely used in natural language processing and information retrieval for text analysis. The TF-IDF score is calculated based on the frequency of a term in a document and its rarity across the entire document collection. This technique aims to highlight terms that are both frequently occurring within a document and unique to that document. Let D represent the set of movie reviews and T represent the set of unique terms in the corpus. The TF-IDF score for term t in document d is computed as follows: TF-IDF(t,d) = TF(t,d)×IDF(t) where TF(t,d) means term frequency given as TF(t,d) = Total number of terms in document d / Number of times term t appears in document d. IDF(t) means inverse document frequency given as IDF(t) = log(Number of documents containing term t / Total number of documents). The TF-IDF parameters include the use of a specific tokenizer and the exclusion of common stop words for enhanced feature extraction

B. BERT Tokenizer

The BERT (Bidirectional Encoder Representations from Transformers) tokenizer is a crucial component of BERT-based models for natural language processing tasks. BERT, developed by Google, is known for its ability to understand the context and relationships between words in a bidirectional manner, capturing intricate language patterns. The tokenizer plays a key role in breaking down input text into smaller units, or tokens, that the model can process. It employs a tokenization technique known as WordPiece, which involves breaking down words into subword units. The process involves iteratively selecting and adding the most frequent subword unit until a predefined vocabulary size is reached. This approach is effective in handling rare or out-of-vocabulary words by representing them as combinations of subword units. Let Psubword(s) represent the probability of selecting a subword unit s during the tokenization process. The final vocabulary V consists of subword units that have been selected based on their frequency in the training data. Given a word W, the tokenization operation involves identifying a sequence of subword units = S=$\{s_1, s_2, s_3, \ldots, s_n\}$such that W can be represented as a concatenation of these subword units: W = $s_1 + s_2 + s_3 + \ldots + s_n$

C. Decision Tree

Decision Tree Analysis is a popular machine learning algorithm that is used for both classification and regression tasks. It models decisions based on a tree-like graph structure, where each internal node represents a decision based on the value of a particular feature, each branch represents the outcome of that decision, and each leaf node represents the final decision or outcome. Root node is the topmost node in the tree, representing the initial decision. Internal nodes are decision nodes that split the data based on a feature. Branches represent the outcomes of the decisions. Leaf nodes are terminal nodes that provide the final decision or prediction. At each internal node, a decision is made based on the value of a specific feature. This process continues until a leaf node is reached, where the final decision or prediction is made. The decision tree algorithm selects the best feature to split the data at each internal node. Common criteria for measuring the effectiveness of a split include Gini impurity, entropy, or mean squared error, depending on whether the task is classification or regression. In classification tasks, Gini impurity measures the likelihood of misclassifying a randomly chosen element. The goal is to minimize Gini impurity during the tree-building process. $Gini(t) = 1 - \sum_i^c p(i|t)^2$ where t is a node, c is the number of classes, and p(i—t) is the probability of class i at node t. Entropy is another measure used in classification tasks, capturing the impurity or disorder in a set of labels. $Entropy(t) = -\sum_i^c p(i|t) log_2(p(i|t))$. Decision Tree Analysis is a versatile and widely used algorithm that forms the basis for more complex ensemble methods like Random Forests. Its interpretability and flexibility make it a valuable tool in various machine learning applications

D. Multinomial Naive Bayes

Multinomial Naive Bayes is a variant of the Naive Bayes algorithm that is specifically designed for text classification problems where the features (attributes) are discrete and represent the frequency of words or other categorical features. It's particularly well-suited for tasks like document classification and spam filtering. In text classification, the features are typically the word frequencies or word counts in a document. The document is represented as a bag-of-words, ignoring the order and structure but keeping track of the frequency of each word. The multinomial distribution is used to model the probability distribution of a set of categorical features. In the context of Multinomial Naive Bayes, the categories are the possible values of the features (e.g., word occurrences). Let's consider a document with features $X = \{x_1, x_2, x_3, ..., x_n\}$ where x_i represents the frequency of word i in the document. The class variable is denoted as C. The probability of a class C given the document's features X is calculated using Bayes' Theorem and the multinomial probability distribution: $P(C|X) \propto P(C).\Pi_{i=1}^n P(x_i|C)$.The probability $P(x_i|C)$. It's often modeled as: $P(x_i|C)$ = (count of occurrences of word i in documents of class C+α)/(total count of words in documents of class C+α.vocabulary size). Here α is a smoothing parameter which is used to handle the issue of zero probabilities when a word does not occur in a particular class during training. The vocabulary size represents the total number of unique words in the entire training dataset. During the training phase, the algorithm estimates the probabilities P(C) and $P(x_i|c)$ from the training data. It calculates the word frequencies for each class and uses them to estimate the parameters of the multinomial distribution. During the prediction phase, the algorithm calculates the posterior probability for each class given the document's features and assigns the class with the highest probability as the predicted class.

E. *Bidirectional Encoder Representations from Transformers*

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art natural language processing (NLP) model developed by Google. It was introduced in the paper titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Jacob Devlin and his colleagues at Google Research. BERT has achieved significant success in various NLP tasks and has become a landmark model in the field. BERT takes tokenized input sentences and represents them as vectors. Each token is embedded into a high-dimensional vector space. Special tokens, like [CLS] (for classification) and [SEP] (to separate sentences in a pair), are added to the input. BERT employs multiple layers of transformer encoders. Each encoder layer consists of self-attention mechanisms and feedforward neural networks. The output of each layer is a contextualized representation of the input tokens. The self-attention mechanism in BERT allows each token to attend to all other tokens in the input sequence, capturing dependencies and relationships. The attention mechanism computes a weighted sum of the values (output of the previous layer) based on attention scores. The attention score $A_{ij}^{(l)}$ for the i-th token attending to the j-th token in the n-th layer is computed as : $A_{ij}^{(l)} = softmax((Q_i^{(l)}.(k_j^{(l)})^T)/(\sqrt{d_k}$ where $Q_i^{(l)}, K_j^{(l)} and V_j^{(l)}$ are the query the query, key, and value representations for tokens i and j in layer l. d_k is the dimensionality of the key vectors. The output of each transformer layer is computed as follows:$Output^{(l)}$=$LayerNorm(Attention(Input^{(l)}) + (Input^{(l)})$. Here LayerNorm denotes normalization. BERT is pre-trained on two unsupervised tasks : Masked Language Model (MLM): Predicting masked words in a sentence $P(w_i|context)$. Next Sentence Prediction (NSP): Determining whether two sentences follow each other.P(IsNextSentence 1,Sentence 2). After pre-training, BERT can be fine-tuned on downstream tasks. For classification tasks, the [CLS] token representation is used as the aggregate sequence representation, and a softmax layer is applied for classification. While these are some of the key mathematical components, BERT's architecture is quite sophisticated, and the actual model involves many hyperparameters, layer-wise interactions, and other details that make a comprehensive mathematical representation complex. The above provides a high-level understanding of the main mathematical concepts underlying BERT.

III. Experiments

A. *Decision Tree Classifier*

The first experiment conducted on the IMDB spoiler dataset is Decision Tree analysis. It is performed by initially loading 10000 rows for demonstration. The dataset is divided into training and test data sets with test_size set as 0.2. As part of data preprocessing, the Term frequency-Inverse Document Frequency vectorization is performed by setting max_features value set as 10000. TF-IDF vectorization is performed to convert textual data to TF-IDF vectors. The train vectors and test vectors are then collected. The decision tree classifier model is trained on the training data with max_depth value set as 8 for experimentation. The predictions are then made on the test data based on the learned patterns on the training data. One of the main challenges encountered here is the overfitting of the data. The other challenge was the implementation of TF-IDF vectorization in data pre processing. It was a complex process as its impact on the model performance was carefully considered. Then the accuracy score, classification report, confusion matrix, precision recall curve and receiver operating characteristics curve are then plotted to analyse the performance of the decision tree classifier model.

B. *Multinomial Naive Bayes Classifier*

The second experiment conducted on the IMDB spoiler dataset is Multinomial Naive Bayes Classifier analysis. It is performed by initially loading 10000 rows for demonstration. The dataset is divided into training and test data sets with test_size set as 0.2. As part of data preprocessing, the Term frequency-Inverse Document Frequency vectorization is performed by setting max_features value set as 10000. TF-IDF vectorization is performed to convert textual data to TF-IDF vectors. The train vectors and test vectors are then collected. The multinomial naive bayes classifier model is trained on the training data for experimentation. The predictions are then made on the test data based on the learned patterns on the training data. One of the main challenges encountered here also is the overfitting of the data. The other challenge was the implementation of TF-IDF vectorization in data preprocessing. It was a complex process as its impact on the model performance was carefully considered. Then the accuracy score, classification report, confusion matrix, precision recall curve and receiver operating characteristics curve are then plotted to analyse the performance of the multinomial naive bayes classifier model.

C. *Custom built Bidirectional Encoder Representations from Transformers (BERT) model*

The third experiment conducted on the IMDB spoiler dataset is the custom built BERT analysis model. We have constructed a custom neural network architecture on top of the BERT model. It includes a pooling strategy (average pooling) applied to the BERT outputs to reduce sequence length, aggregated using tf.reduce_mean along the axis of sequence length. A dense layer with 128 units, ReLU activation function, and L2 regularization (kernel_regularizer) to extract high-level features from the pooled output.A dropout layer with a dropout rate of 0.5 to prevent overfitting. An output layer with a single unit and sigmoid activation, suitable for binary classification tasks. This final layer outputs the probability of the positive class. The complete architecture is assembled into a model named model_custom_bert using the Keras functional API. The experimentatin is performed by randomly sampling the dataframe. The dataset is divided into training and test data sets with test_size set as 0.3. As part of data preprocessing, the BERT tokenizer is employed for tokenization and encoding the training and test dataframes.

The training and test labels are converted to tensorflow tensors. The custom neural network architecture based BERT model is built by first obtaining last hidden state. Then a pooling strategy is applied to reduce sequence length, modifies based on the needs. A dense layer with 128 units, ReLU activation, and L2 regularization is constructed. One of the main challenges encountered here also is the overfitting of the data. So a dropout layer with a dropout rate of 0.5 is constructed to prevent overfitting problem. An output layer with a single unit and sigmoid activation is built for binary classification. The custom BERT model is then built with the designed input and output layers. The Adam optimizer from keras is defined with a specified learning rate of 1e-5. The custom BERT model is compiled with binary crossentropy loss and with metric as accuracy. The custom BERT model is trained with early stopping. EarlyStopping callback monitors the validation loss and stops training if no improvement after 'patience' epochs is reached. 'patience' is the number of epochs with no improvement after which training will be stopped. The patience value for early stopping is set as 3 here. A model checkpoint callback is defined. The custom BERT model is then trained with early stopping and model checkpoint features, with number of epochs set as 3. The model is then used for predictions on the test data. Then the accuracy score, classification report, confusion matrix, precision recall curve and receiver operating characteristics curve are then plotted to analyse the performance of the custom built BERT model.

D. *User Interface for spoiler detection system*

A user interface is built for the system using flask, a web framework for python, to create an interactive web application. The flask interface is integrated to show the responsive output of custom BERT model. These are the features of the flask web application:
Spoiler Detection Model Integration: Incorporates a spoiler detection model loaded from 'models/checkpoint.h5' and a corresponding tokenizer loaded from BertTokenizer.
User Input and Prediction: Users can input movie reviews through a web form, and the application predicts whether the review contains spoilers or not using the loaded model.
Responsive UI with HTML Templates: Implements a responsive user interface with an HTML template ('index.html') that allows users to enter text and receive real-time predictions.
Dynamic Result Display: Dynamically updates the web page to display the entered text and the model's prediction result (Spoiler or No Spoiler) based on the user input.

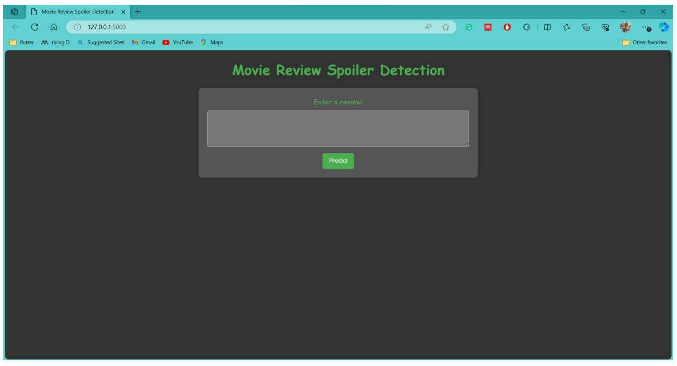

Movie Review Spoiler Detection
127.0.0.1:5000
Movie Review Spoiler Detection
Enter a review:
Predict

Fig. 1. User interface for showing if a movie review contains spoiler or not

IV. Results

The dataset used for experimentation is the IMDB spoiler dataset : https://www.kaggle.com/datasets/rmisra/imdb-spoiler-dataset. It contains meta-data about items as well as user reviews with information regarding whether a review contains a spoiler or not.There are 2 files, namely IMDB_movie_details.json and IMDB_reviews.json. For training the model, the data is used from IMDB_reviews.json file. There are a total of 573,913 rows and 7 columns namely is_spoiler, movie_id, rating, review_date, review_summary, review_text, user_id.The categorical columns are movie_id, rating, review_date and user_id.The text columns are review_summary and review_text. The column is_spoiler is the target variable.

A. *Exploratory Data Analysis*

The exploratory data analysis is performed on the dataset, where the count plots of all categorical columns vs is_spoiler(target variable) are explored.

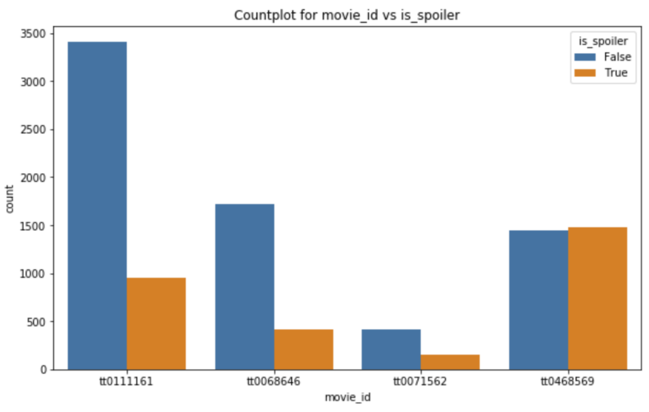


Fig. 2. Count plot for movie_id vs is_spoiler

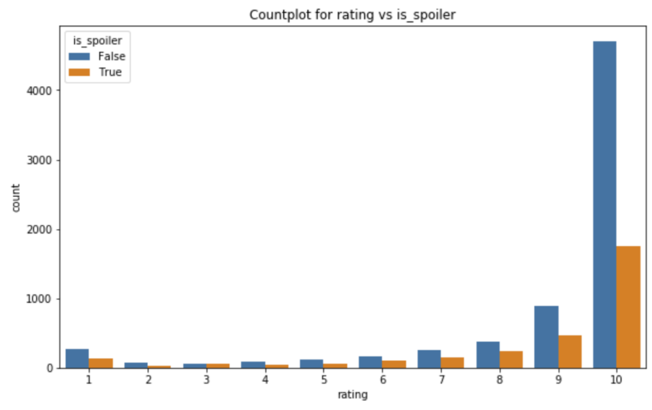


Fig. 3. Count plot for rating vs is_spoiler

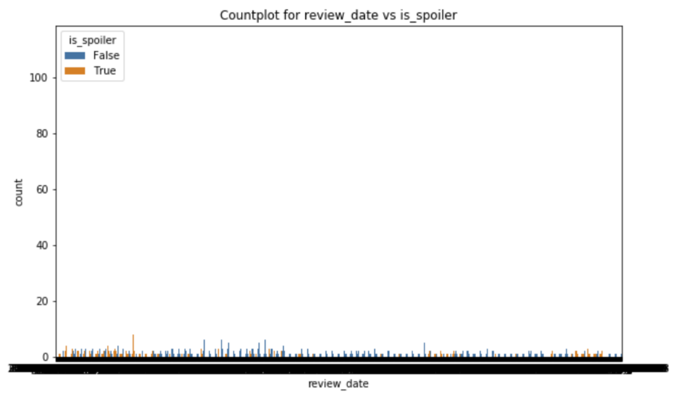


Fig. 4. Count plot for review_date vs is_spoiler

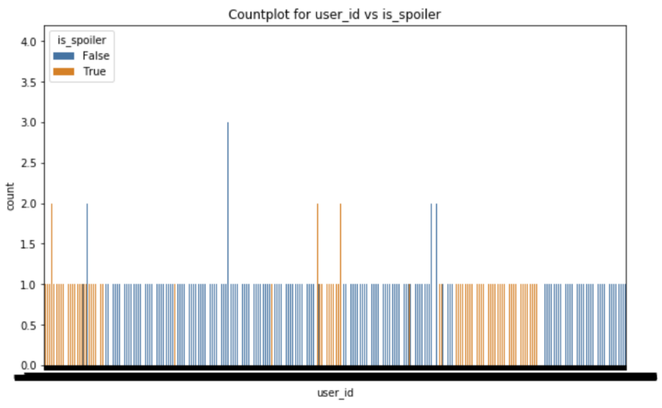


Fig. 5. Count plot for user_id vs is_spoiler

Word clouds for text columns review_summary and review_text for spoiler and non-spoiler reviews are plotted.

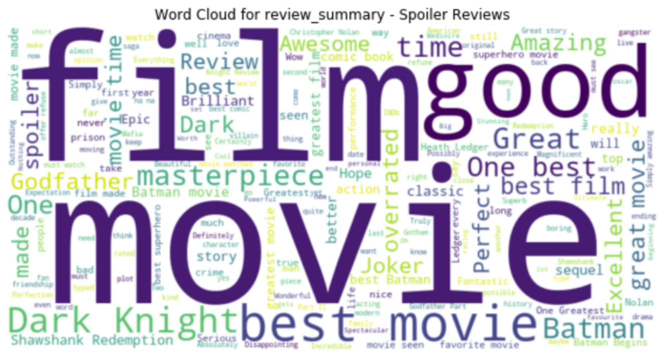


Fig. 6. Word cloud for review_summary vs Spoiler reviews

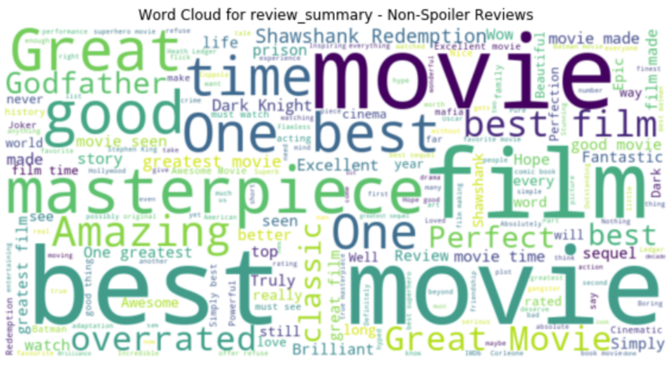


Fig. 7. Word cloud for review_summary vs Spoiler reviews

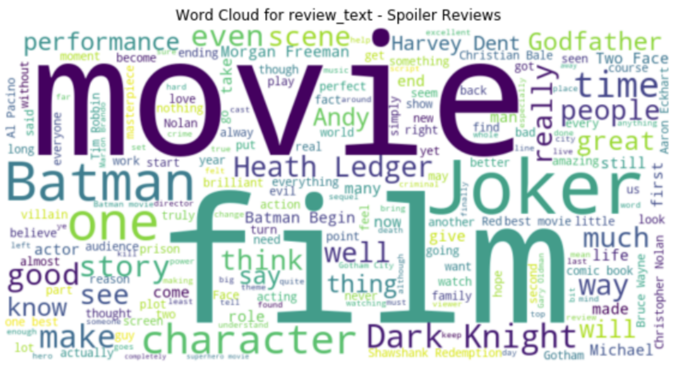


Fig. 8. Word cloud for review_text vs Spoiler reviews

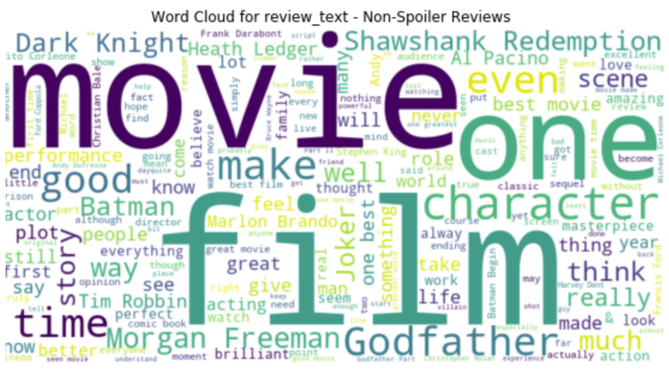


Fig. 9. Word cloud for review_text vs Spoiler reviews

Pie charts are depicted for spoiler distribution within the reviews.

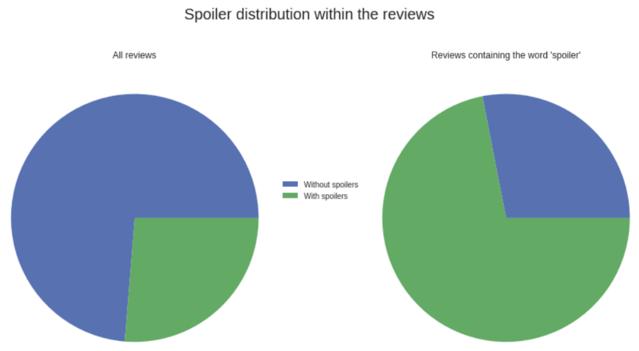


Fig. 10. Spoiler distribution within the reviews

The distribution of spoiler Vs no spoiler reviews according to the review length and the distribution of spoiler ratio Vs ratings given by the user are explored. It is inferred that reviews_text and is_spoiler are the columns needed to be used for training the model.

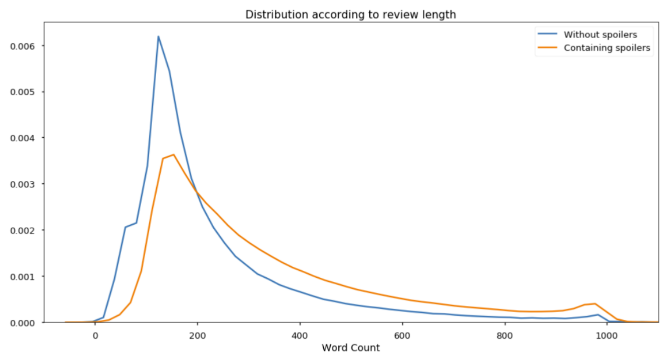


Fig. 11. Distribution of spoilers vs no spoiler reviews according to word length

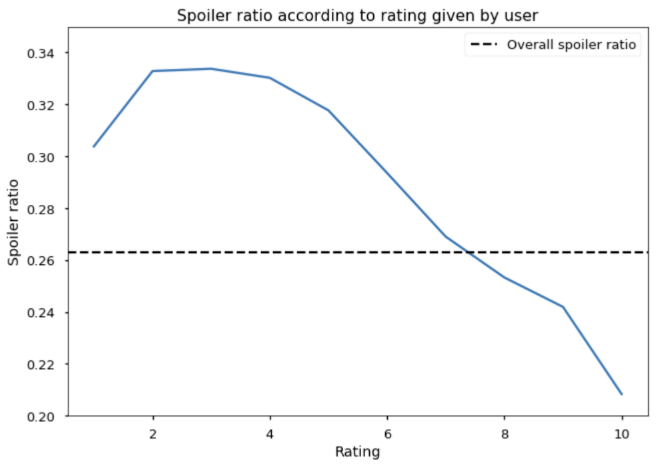
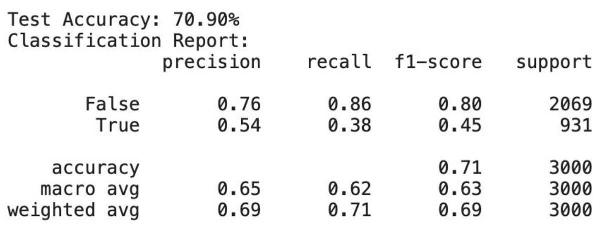


Fig. 12. Spoiler ratio according to rating given by user

B. *Decision Tree Classifier Results*

This model returned a test accuracy of 70.90% and area under curve value of 0.62.Here are the classification report, confusion matrix, precision-recall curve and the receiver operating characteristics curve.



```
Test Accuracy: 70.90%
Classification Report:
              precision    recall  f1-score   support

       False       0.76      0.86      0.80      2069
        True       0.54      0.38      0.45       931

    accuracy                           0.71      3000
   macro avg       0.65      0.62      0.63      3000
weighted avg       0.69      0.71      0.69      3000
```

Fig. 13. Decision Tree Classifier model classification report

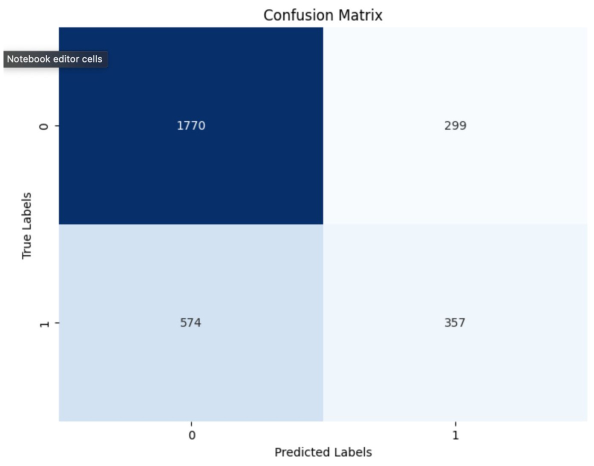


Fig. 14. Decision Tree Classifier model confusion matrix

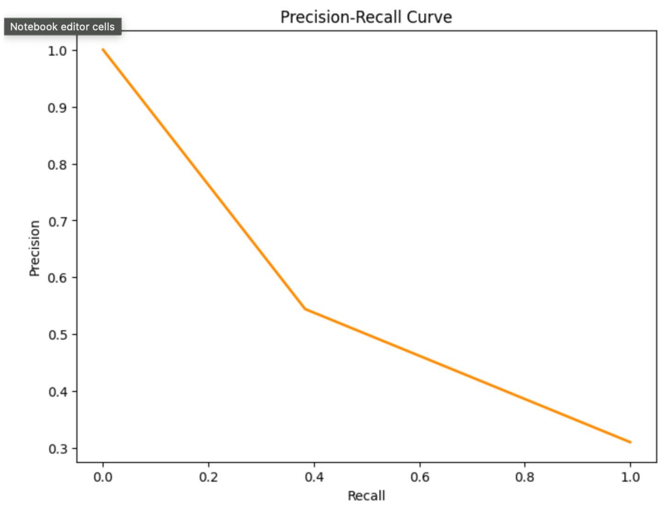


Fig. 15. Decision Tree Classifier model Precision Recall Curve

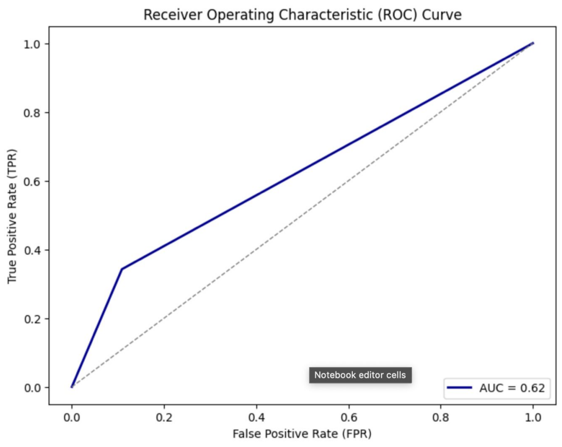


Fig. 16. Decision Tree Classifier model ROC Curve

C. *Multinomial Naive Bayes Classifier Results*

This model returned a test accuracy of 71.67% and an average precision value of 0.37. The model returned an area under curve value of 0.56.Here are the classification report, confusion matrix, precision-recall curve and the receiver operating characteristics curve.

```
Test Accuracy: 71.67%
Classification Report:
              precision    recall  f1-score   support

       False       0.72      0.97      0.83      2069
        True       0.70      0.15      0.25       931

    accuracy                           0.72      3000
   macro avg       0.71      0.56      0.54      3000
weighted avg       0.71      0.72      0.65      3000
```

Fig. 17. Multinomial Naive Bayes model classification report

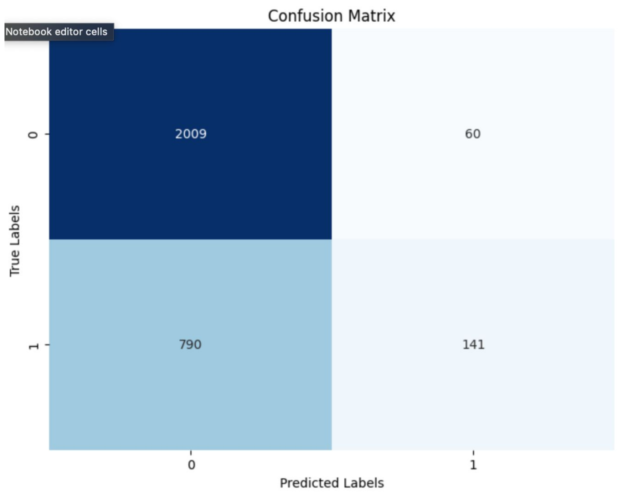


Fig. 18. Multinomial Naive Bayes model confusion matrix

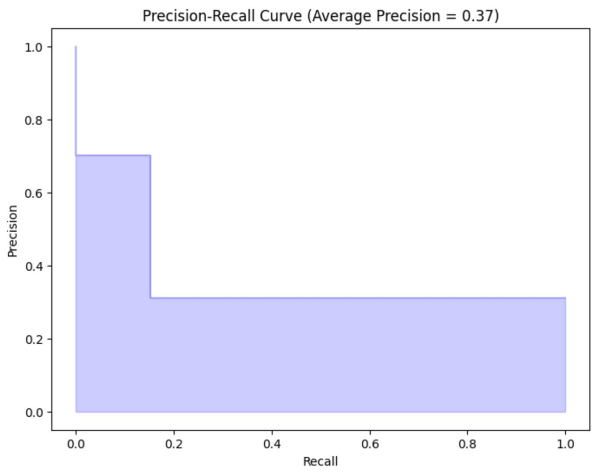


Fig. 19. Multinomial Naive Bayes model Precision Recall Curve

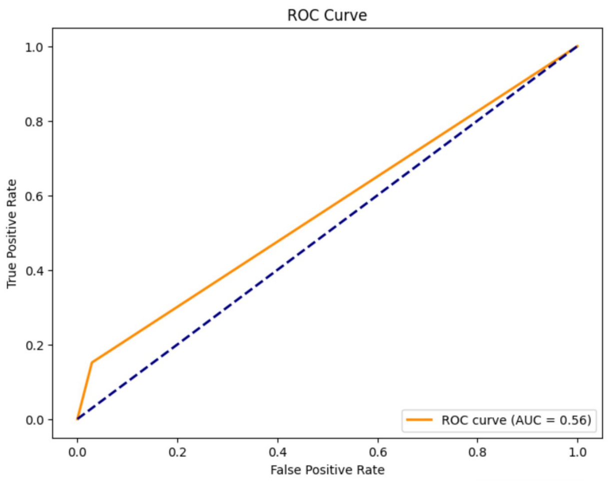
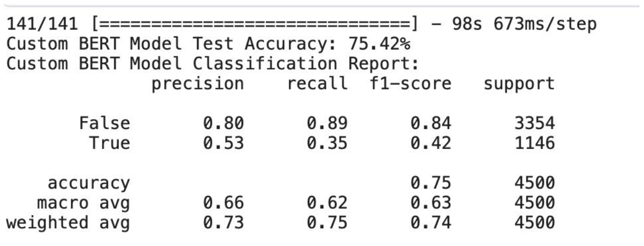


Fig. 20. Multinomial Naive Bayes model ROC Curve

D. Custom built BERT Model

This model returned a test accuracy of returned a test accuracy of 75.42% and an area under curve value of 0.62. Here are the classification report, confusion matrix, precision-recall curve and the receiver operating characteristics curve.



```
141/141 [==============================] - 98s 673ms/step
Custom BERT Model Test Accuracy: 75.42%
Custom BERT Model Classification Report:
              precision    recall  f1-score   support

       False       0.80      0.89      0.84      3354
        True       0.53      0.35      0.42      1146

    accuracy                           0.75      4500
   macro avg       0.66      0.62      0.63      4500
weighted avg       0.73      0.75      0.74      4500
```

Fig. 21. Custom BERT model classification report

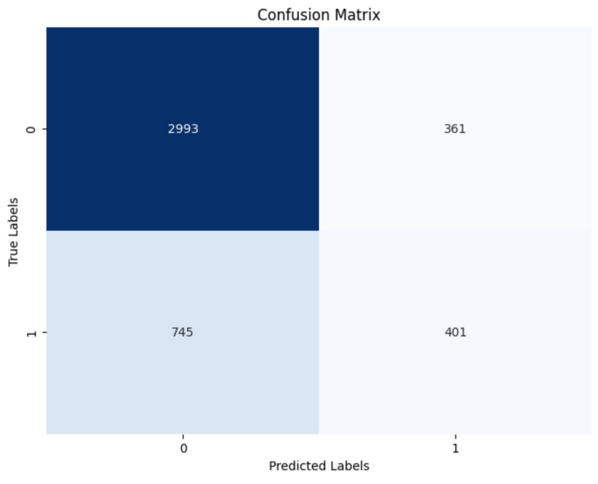


Fig. 22. Custom BERT model confusion matrix

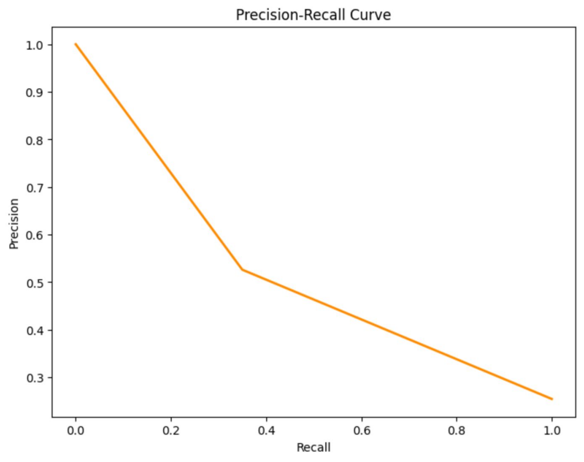


Fig. 23. Custom BERT model Precision Recall Curve

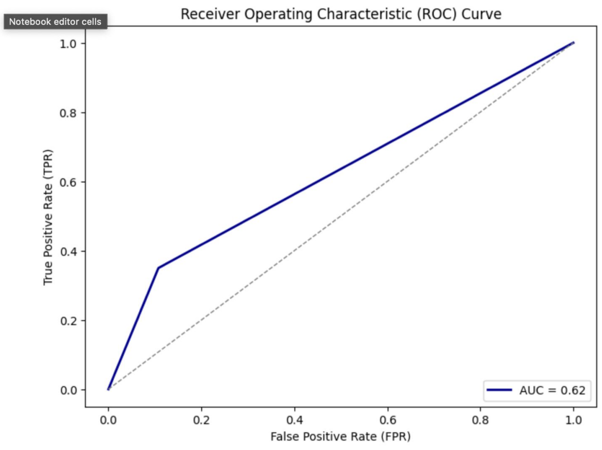
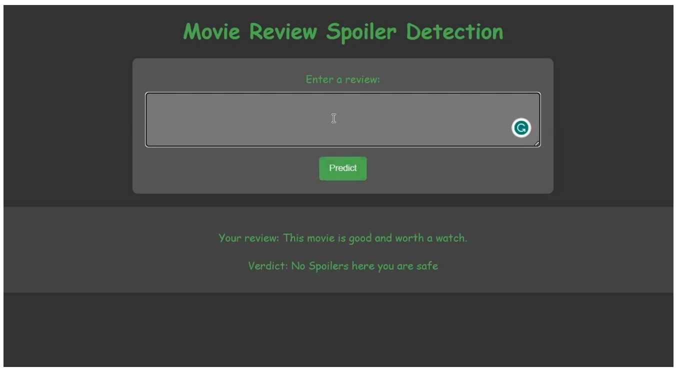


Fig. 24. Custom BERT model ROC Curve

E. Spoiler and non spoiler movie review predictions on user interface

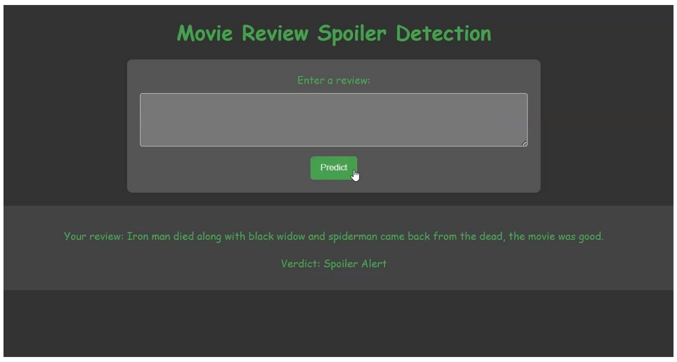
Here are the figures showing the results on user interface when a movie review not containing any spoilers is typed in and when a movie review containing a spoiler is typed in.



Your review: This movie is good and worth a watch.

Verdict: No Spoilers here you are safe

Fig. 25. Non Spoiler review prediction shown on UI



Your review: Iron man died along with black widow and spiderman came back from the dead, the movie was good.

Verdict: Spoiler Alert

Fig. 26. Spoiler review prediction shown on UI

V. Conclusion

While employing the machine learning models, overfitting was one of the major problems faced during training. The implementation of TF-IDF vectorization was a little complex process. Training the deep learning model of custom BERT was computationally expensive, the early stopping and model check points helped in preventing model from overfitting. The custom neural network architecture of BERT based model involves hyperparameters like the number of units in the dense layer, dropout rate, and L2 regularization strength which were complex to implement.

Focusing on benefits an limitations of the system, improved user experience, content engagement, personalized recommendations, enhanced content accessibility are few of the benefits that can be observed by the development of the system. Where as contextual understanding of words, false positives and false negatives, variability in user sensitivity, resource intensiveness of BERT model are few of the limitations observed in the system.

In conclusion, exploratory data analysis is performed on the imdb spoiler detection dataset, where review_text and is_spoiler columns are known to be useful for model training. A total of 3 experiments were ran on the imdb spoiler detection dataset, the models involved includes decision tree classifier, multinomial naïve bayes classifier and custom built BERT model. The decision tree classifier performed the least, followed by multinomial naïve bayes classifier model.The custom built BERT model performed better among all the three models with a test accuracy of 75.42%. A web application was built using flask web framework in python. This application dynamically updates the web page to display the entered text and the model's prediction result (Spoiler or No Spoiler) based on the user input.

References

Boyd-Graber, J., Glasgow, K., & Zajac, J. S. (2013). Spoiler alert: Machine learning approaches to detect social media posts with revelatory information. *Proceedings of the American Society for Information Science and Technology*, *50*(1), 1–9.

Jeon, S., Kim, S., & Yu, H. (2016). Spoiler detection in TV program tweets [Special issue on Discovery Science]. *Information Sciences*, *329*, 220–235.

Lindo, A. (2020). *Movie Spoilers Classification Over Online Commentary, Using Bi-LSTM Model With Pre-trained GloVe Embeddings* [Doctoral dissertation, Dublin, National College of Ireland].

Wan, M., Misra, R., Nakashole, N., & McAuley, J. (2019). Fine-grained spoiler detection from large-scale review corpora. *arXiv preprint arXiv:1905.13416*.

Wang, H., Zhang, W., Bai, Y., Tan, Z., Feng, S., Zheng, Q., & Luo, M. (2023). Detecting Spoilers in Movie Reviews with External Movie Knowledge and User Networks.