Identification of Spoiler in IMDB movie review

Mowniesh Asokan (mowas455)

August 2022

Abstract

This report demonstrates an NLP-based approach to detecting spoilers in the IMDB review. Generally, these reviews reveal some information associated with the plot of a movie. An automated system, filtering out such spoilers, would be ideal as manual labeling is impossible due to a large amount of content. To identify those reviews, we explore machine learning and deep learning-based approaches. We use techniques Bi-LSTM, XGBoost, Random Forest, and Naive Bayes, and test the efficiency of these models in identifying the spoiler-based reviews. We also created feature extractors using word embeddings (Word2vec & Glove), cosine similarity, and Term-Frequency and Inverse Document Frequency (TF-IDF) method. We perform a quantitative analysis based on accuracy metrics to assess the aforementioned models.

Keywords: Bi-LSTM, GloVe, NLP, Spoiler, word2vec, Word embeddings, semantic approach.

Contents

Chapter 1

Introduction

Spoilers can destroy the experience of watching a movie. Some reviews are written essays in which IMDb users explain what they liked or disliked about a movie and offer other criticism. Review can vary in length from a few lines to several hundred words. Those reviews may sometimes ruin the enthusiasm towards the scenes. IMDB and similar sites allow users to mark articles and posts as spoilers, but what if nobody does? Our goal was to check if there was some way to search out spoilers without looking forward to the author of the post to mark it. Our work involves observing all aspects. We can use movie details, descriptions. But most of the movie review text itself is employed for identification. An identical application is the Twitter Sentiment projects, which identify positive and negative tweets. Similarly, our project tries to classify posts as spoilers instead of sentiments.

As spoiler annotations are not supported universally and require intense work and time. Due to these hindrances, automating this task would be crucial. The task of detecting spoilers is nontrivial. as example, the knowledge of a character dying may be a part of the premise of a story and, therefore, not a spoiler, but may additionally constitute a serious turn within the plot. At the same time, it seems likely that a straightforward heuristic, e.g., one trying to find the word 'killed,' would have a point of success at identifying spoilers.

We will use deep learning as well as traditional machine learning approaches, fine-tuning them using user-generated annotations to predict which documents contain spoilers. On the data, we'll also build a model that predicts the presence of spoilers on a token level, thereby performing the sentence level classification to detect the spoilers.

1.1 Motivation

The motivation of this project is to build a novel supervised machine learning (ML) model to improve the accuracy of text classification in movie reviews. Thus, improving the accuracy of detecting spoilers within the text to counterbalance the spoiler-phobia phenomenon that spreads through all these modern online communities. At the same time, that pursues providing trustworthy work that serves as the foundation for future efforts against spoilers as a deal-breaker for global culture and the film industry.

Chapter 2

Data

The data came from a Kaggle post by Rishabh Misra, which can also be found here [11]. Overall, there were 573,913 reviews covering 1,572 different movies. Among these, 150,924 reviews were found to have spoilers. This dataset came in two JSON files with reviews and movies. It contained the review date, the movie id, user id, spoiler tag, review, rating, and user summary. A movie file contained information about the title, id, plot summary, duration, genre, rating, and plot synopsis for a movie.

2.1 Preparation & Pre-procesing

Our first job is to prepare the dataset to put in the form of usable format. Then combine the two datasets to extract useful information in movie details for predicting the spoilers. The next step is to convert all other columns into numerical features and drop superfluous columns for the ID of the movie and the use to arrive at our working dataset. From the exploratory analysis of the review text, we find that some reviews contain a word spoiler. Therefore, it also may help to find the spoiler text in the reviews.

We took the main feature for classification as review text and review summary. To work with the text, we need to change the text into a consistent format by removing the stop words, symbols such as questions, and exclamation marks in the review text. This transforms

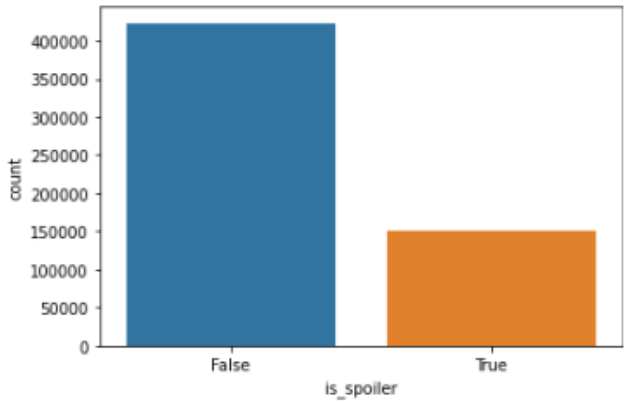

Figure 2.1: The complete distribution of spoiler data

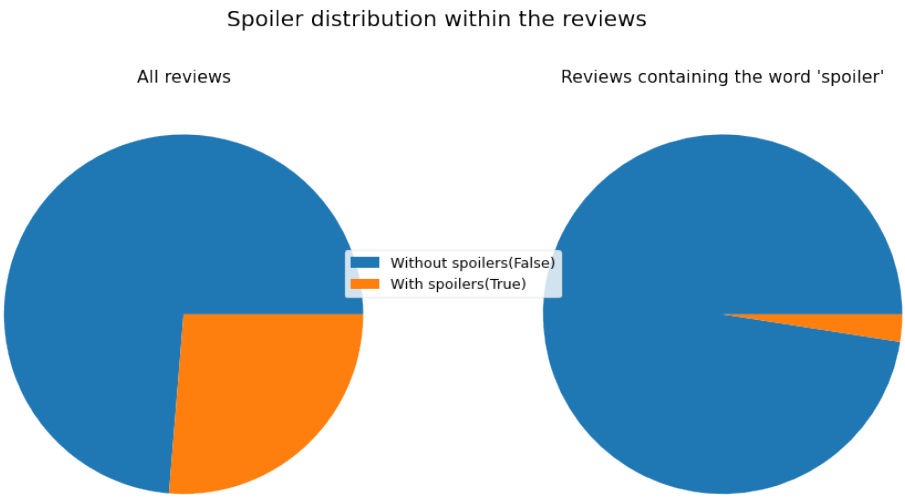

Figure 2.2: Sample movie spoilers in the review

the data into a more efficient format for modeling. In this stage we perform normalization and transformation of text helps to clean the unstructured text for sentiment analysis, then tokenization of the text into word tokens to identify the patterns and boundaries between every word in the text corpora. In figure 2.1, the bar plot shows the complete distribution spoiler in the data set. The plot, clearly shows that the distribution of the data is not equal. So we used 110000 observations with an equal number of spoilers and non-spoilers filtered.

Therefore, all the pre-process actions will help the models classify the review text as a spoiler or non-spoiler. Since we are dealing with a large amount of text corpus with millions of words, so there will be a lot of text preprocessing techniques to remove the unwanted noisy text from the data.

Chapter 3

Theory

This chapter presents the theory corresponding to the background information required for understanding the techniques used in later chapters. A particular focus of our discussion is the RNN architecture of Bi-LSTM, XGBoost, Random Forest, and Naive Bayes models. We choose traditional metrics for evaluating machine learning models.

3.1 RNN-LSTM

When we have a small RNN[6], we would be able to effectively use the RNN because there is no problem with vanishing gradients. But, when we consider using long RNNs there is not much we could do with the traditional RNN and hence it wasn't widely used. That is the reason that leads to the finding of LSTM [7] which basically uses a slightly different neuron structure. In LSTM, we will be referring to a neuron as a cell and it also works on the **"Gating Mechanism"**. Which regulates the information that the network stores if it has to pass the information to the next layer or forget the information it has in memory.

3.1.1 Bi-LSTM:(Bi-directional Long Short Term memory)

Bi-direction recurrent neural networks(RNN) [10] are putting two independent RNNs together. This structure allows the network to have both backward and forward information about the sequence at every time step. Using bidirectional will run our inputs in two ways, one from past to future and one from future to past, and what differs this approach from unidirectional is that the LSTM runs backward to preserve information from the future and using the two hidden states combined in any point in time to preserve information from both past and future.

3.2 XGBoost

eXtreme Gradient Boosting was proposed by Tianqi Chen in 2015 and is one of the boosting algorithms. It is proved in the literature [4] that the XGBoost model has the characteristics of low computational complexity, fast running speed, and high accuracy. The idea of the Boosting algorithm is to integrate many weak classifiers into a strong classifier. Because

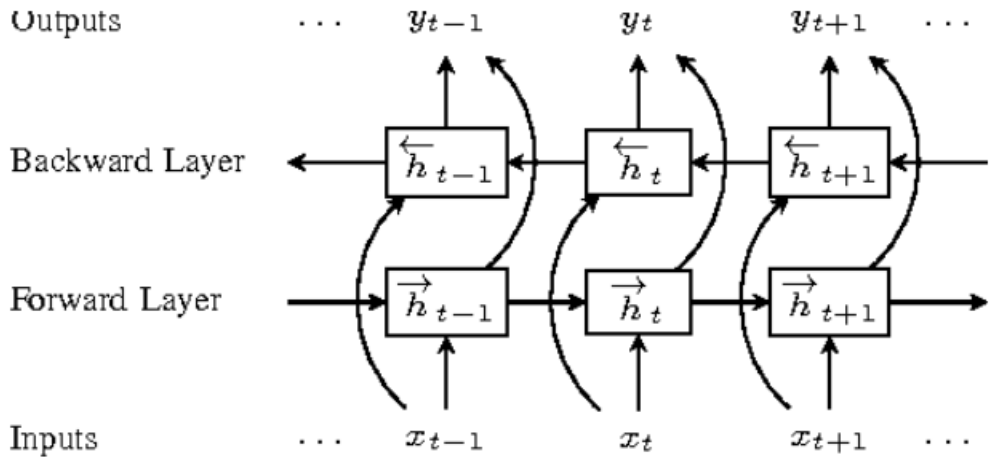

Figure 3.1: : BI-LSTM Architecture

XGBoost is a lifting tree model, it integrates many tree models to form a strong classifier. The tree model used is the CART (Classification and Regression Tree) model.

System Optimization:

1. Parallelization: XGBoost approaches tree building sequentially using a parallel implementation. This is because base learners can be built using interchangeable loops. The outer loop, which enumerates the leaf nodes of a tree, and the inner loop, which calculates the features, work together. Parallelization is limited by nesting loops because without completing the inner loop (the more computationally intensive of the two), the outer loop cannot begin. In order to improve runtime, the order of loops is exchanged through initialization, which uses a global scan of all instances and sorting via parallel threads. As a result, the switch enhances algorithmic performance by reducing parallelization overheads.

2. Tree Pruning: Tree splitting within the GBM framework is greedy by nature and is based on a negative loss criterion at the point of splitting a tree. As specified, XGBoost starts pruning trees backward using the 'max depth' parameter instead of the criterion. This 'depth-first' approach improves computational performance significantly.

3. Hardware Optimization: This algorithm has been designed to make efficient use of hardware resources. This is accomplished by allocating internal buffers for storing gradient statistics within each thread. Furthermore, there are enhancements such as 'out-of-core' computing that maximize available disk space while handling big data frames that cannot fit in memory.

3.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at

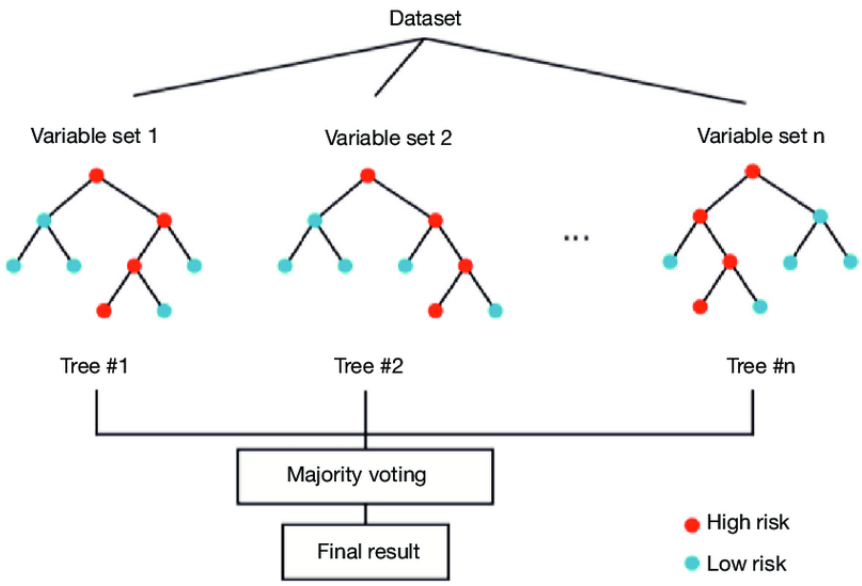

Figure 3.2: The architecture of the Random Forest model

training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees citation needed. However, data characteristics can affect their performance.

Random forest [3], as its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

3.4 Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks and text categorization [5]. The crux of the classifier is based on the Bayes theorem. which is our **baseline model** for the spoiler identification task.**Bernoulli Naive Bayes:**This is similar to the multinomial naive Bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only binary values, for example, if a spoiler is in the text or not. The Bayesian equation and the decision rule for Bernoulli naive Bayes are shown below,

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{3.1}$$

$$P(x_i|y) = P(i|y)x_i + (1 - p(i|y))(1 - x_i) \tag{3.2}$$

Chapter 4

Methods

In this section, we illustrate the methods and ideas used for the binary classification of spoiler detection, such as Naive Bayes Classifier (NB), XG-Boost, Random Forest, and Bidirectional Long Short-Term Memory Bi-LSTM with a pre-trained Glove and word2vec embedding model[1]. For a machine learning model, the text should be transformed into numerical sequences to work with the calculation of prediction. By selecting the well-known and established techniques for transforming string into numerical vectors of words. Therefore, the numerical vector act as input for the machine learning models.

We have chosen to utilize the following word representation techniques: Global Vectors (Glove), Term Frequency Inverse Document Frequency (TF-IDF), and Continuous Bag of Words (CBOW). To represent text documents for Bi-LSTM, NB, and XGBoost models. We implemented this research using the Python 3.7 programming language due to its simplicity and readability. kaggle Notebook computational power with GPUs on the Jupyter Notebook.

4.1 TF-IDF

Term frequency-inverse document frequency is a text vectorizer that transforms the text into a usable vector. It combines two concepts, Term Frequency (TF) and Document Frequency (DF). The term frequency represents the number of occurrences of the specific term in the document, and document frequency is the number of documents containing a particular word. TF-IDF is just a multiplication of the term frequency and inverse document frequency. Inverse document frequency (IDF) is the weight of a term it aims to reduce the weight of a term if the term's occurrences are scattered throughout all the documents.

$$idf_i = log\frac{n}{df_i} \tag{4.1}$$

$$TF - IDF = tf_{i,j} + idf_i \tag{4.2}$$

4.2 Word Embedding

Word embedding is typically represented using real-valued vectors that encode the meaning of a word so that the words that are close in vector space should have a similar meaning [14].

The embedding captures semantic relationships, only when they are trained on a huge text corpus, using some language modeling approach. Before training, the word embedding is randomly initialized, and they did not make any sense. It is only when the model is trained, therefore that the word embedding has captured the semantic meaning of all the words.

4.2.1 Semantic-based approach

Semantic-based approaches deal with the relationship between words. One of the best things that happened in Natural Language Processing is Word2Vec. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector[2]. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors. In this project, this experiment uses the Semantic relations between word vectors and determines the similarity between two documents to solve the spoiler identification problem.

4.2.2 Word2vec embeddings

Word2vec is one of the popular techniques to learn word embedding using a neural network model. Word embedding via word2vec can make the natural language model implement the mathematical operation on the words to detect the spoiler in the reviews[8]. There are two types of models for word2vec - (CBOW) Continuous Bag of Word and Skip-gram model. Gensim word2vec requires a format of 'list of lists' for training where every document is contained in a list and every list contains lists of tokens of that document. Particularly, we used the continuous bag of words for the word embedding.

Continuous Bag of Words(CBOW): The CBOW model learns to predict a target word in its neighborhood, using all words. To predict the target word, the sum of the background vectors is used. A pre-defined window size surrounding the target word defines the neighboring terms that are taken into account. The CBOW model calculates the average of the context vectors.

Skip-Gram: Skipgram model predicts the context for given a word. The skip-gram model is the exact opposite of the CBOW model. In this case, the target word is fed at the input, the hidden layer remains the same, and the output layer of the neural network is replicated multiple times to accommodate the chosen number of context words. This model works well with a small amount of training data and represents well even rare words or phrases.

4.2.3 GloVe embeddings

GloVe captures both global statistics and local statistics of a corpus. It is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase linear substructures of the word vector space.

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{4.3}$$

Specifically, GloVe [12] will be used together with our Bi-LSTM model. As a result, a vocabulary containing all words in the movie reviews dataset was created. The next step is

to verify whether GloVe recognizes the words in our vocabulary. At first, and before running (pre-processing) tasks, we found GloVe embeddings only for 12 percent of the vocabulary and 90 percent of all reviews. Addressing this issue, we used GloVe pre-trained embedding itself and created a filter based on: a list of common characters (white list), another list containing known special characters recognized by GloVe, and lastly, another one made of all special characters that are not recognized by GloVe. As a consequence, we could filter some noisy characters before starting the post-processing phase. Additionally, during the pre-processing stage, we needed to iterate back and forward, so that we could run additional pre-processing tasks to achieve higher coverage values.

4.3 Evaluation Metric

Since our binary dataset is skewed toward reviews without spoilers, we chose to evaluate our models using AUC. We would see our models as effective if we had used accuracy, but in reality, it might have been just predicting whether a review contains a spoiler or not. Moreover, false positives are more useful than false negatives, since flagging reviews that do not contain spoilers is not as bad as keeping spoiler-filled reviews from being detected by the model.

$$TPR = \frac{TP}{TP + FN} \tag{4.4}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.5}$$

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is the probability that the model ranks a random positive example more highly than a random negative example. When AUC = 1, then the classifier can perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

Chapter 5

Implementation and Result

In our case, we conducted six fields of study comprising the three different techniques, where the model performance results with TF-IDF vectorizer, Genism, and Pretrained-GloVe are shown below.

5.1 Naive Bayes with TF-IDF

Naive Bayes is the first experiment on spoiler identification, and we fix this as a baseline model. We have chosen to train by Bernoulli Naive Bayes Classifier. While using a small amount of data in the training stage, we aim to increase the Recall value per class while maintaining a fast computational ability. Bernoulli NB was specifically designated for this experiment due to the distinct penalization of the non-occurrences rule, where this type of NB algorithm applies over binary features.

5.2 XGBoost with TF-IDF

In this case, we attempted to build a good classifier for our text corpora[13], by appealing to the potential of the eXtreme Gradient Boosting Algorithm. While fitting this data into a machine learning model, requires the data in a more specific state (transformed form) to classify the spoilers. we have chosen the parameter eta is 2 and the objective function is binary: *logitraw*. Moreover, the resulting test accuracy and f1 score are relatively higher than the Naive Bayes model. The Area under the curve (AUC = 0.66) also has some improvement.

5.3 Semantic approach with Review and Plot summary

Word embedding works on the concept of a continuous bag of words and a one-hot vector. It makes Word2Vec is most efficient in terms of determining the similarity by calculating cosine distance. The similarity of text is determined by comparing word vectors or word embedding, multi-dimensional meaning representations of a word. Word vectors are, generated using an algorithm like word2vec. In this approach, a vector of every word in each document is considered a whole, and cosine similarity [9] is calculated between the review text and the original plot summary of every individual movie. A threshold value is a hyperparameter, and

it helps to classify the review text. When threshold value = 0.9, this approach has shown low-performance results.

5.4 Bi-LSTM with Word2vec

In this experiment, We tried the pre-trained word2vec embedding layer with a Bi-LSTM model with a co-occurrence matrix. Before entering into the model, we used a tokenizer to produce a token for a given word and took the maximum length of 400 characters of a review after we simply truncate the input review and then padded the input to the max length of 400. The model is built with the embedding layer using the word2vec weights matrix and a bidirectional layer with 64 units. word2vec only considers the local properties of the dataset. The weight matrix for the embedding layer was formed with help of the google pre-trained word2vec model. They are models which have already been trained on huge amounts of data — billions of words — and use that training as a starting point for new problems. Since we are not training this model ourselves from scratch, the pre-trained model is never completely perfectly tuned for this spoiler identification problem. Even though the performance result of this approach is poor, it slightly better than the previous approaches.

5.5 Random Forest with GloVe

This experiment appealed to the potential of decision trees in order to build a good classifier for our text corpora. In order to test the predictive capabilities of more than one decision tree (a forest of decision trees), we built an RF classifier. Here we used the pre-trained glove embedding to find the embedded weight matrix and with the help of weight or coefficients, each review text is converted into vectors. By giving the array of sentence vectors to a random forest classifier. The prediction performance obtained from the RF classifier is not higher than the Naive Bayes classifier.

5.6 Bi-LSTM with GloVe

The final experiment is Bi-LSTM, we put in place a pre-trained Glove word embedding layer with a co-occurrence matrix. Before entering into the model, we used a tokenizer to produce a token for a given word and took the maximum length of 400 characters of a review after we simply truncate the input review and then padded the input to the max length of 400. The model is built with the embedding layer using the GloVe weights matrix and a bidirectional layer with 64 units. In addition to that stack the two dense layer with *Relu activation* function and the final layer with *Sigmoid activation* function. Finally passing the training data both forward and backward through the LSTM network with (epoch = 10) iteration. The performance result obtained from this method is the most convincing Accuracy and AUC score.

Method	Classification	Precision	Recall	F1 Score	Accuracy	AUC Score
NB-TF-IDF	non-spoiler	0.60	0.77	0.68		
	spoiler	0.68	0.49	0.57	0.63	0.629
XGBoost-TF-IDF	non-spoiler	0.62	0.82	0.71		
	spoiler	0.70	0.50	0.60	0.66	0.66
Bi-LSTM- word2vec	non-spoiler	0.68	0.74	0.71		
	spoiler	0.72	0.64	0.68	0.69	0.69
Semantic similarity	non-spoiler	0.11	0.63	0.19		
	spoiler	0.93	0.51	0.66	0.52	0.52
Random Forest - Glove	non-spoiler	0.62	0.51	0.55		
	spoiler	0.58	0.69	0.63	0.60	0.60
Bi-LSTM- GloVe	non-spoiler	0.68	0.73	0.71		
	spoiler	0.71	0.66	0.69	**0.70**	**0.6974**

Table 5.1: Text classification Model results

Chapter 6

Discussion

By assessing a model's AUC score, one can tell if the model has been successful in positioning random examples to the right side of classification, either positive or negative. AUC can be used to place observations on the right-hand side, where the more True Positive observations that are logged correctly, the closer it is to 1. Meaningless False Positive observations still need to be corrected and classified as False Positives. viewing table 5.1, the Naive Bayes classifier achieved a fair precision score by correctly classifying the class spoiler with a score of 0.68. However, the classifier did not perform as well as the non-spoiler with a precision score of 0.60. Moreover, the AUC score of 0.63 shows that the model has a low separation capacity. For the future approach, it would not be a suitable solution. Likewise, to attain a good prediction, we used a combination of TF-IDF with an XGBoost model. It shows the precision result of spoiler is improved to 0.70 and non-spoiler is to be 0.62, on account of this AUC score of this model also improved compared with NB-Classifier. This lack of prediction is evidentially proven in the confusion matrix result of both models.

To improve the prediction result, we moved to the CBOW technique with the word2vec model. Therefore the padded sequence of a bag of words is trained into the Bi-LSTM model to get a prediction. This method shows the **improvement** in the result of spoiler and a non-spoiler precision score of 0.65, and the resultant AUC score is also 0.69. Therefore the word2vec with the Bi-LSTM model learned the Classification task equally. With the help of the same CBOW technique, we implemented the cosine similarity function to get a similarity of review text and plot summary of the respective movie. To get a prediction on this cosine function, we fixed the threshold as 0.9 or 0.8. If the cosine value of the review text is above the threshold then we name that review a *spoiler*, because the review text reveals the original plot summary of a movie. On the other hand, if it is less then it is *non-spoiler*. Even though it conceptually gives meaning, it does not work fine with the prediction score as shown in table 5.1. While implementing the GloVe, word-to vectors are generated and implemented by the random forest classifier. Even though the RF algorithm is being categorized as robust and versatile with proven effectiveness in predictive modeling. Our RF classifier achieved a poor AUC score of 0.59, hence evidencing a lack of class separation ability

On the other side of the coin, our final model for the classification of movie reviews achieved the highest AUC score out of all the other models. Our Bi-LSTM model using the pre-trained GloVe technique can predict high numbers of True Positives and a fair number of True Negatives. Although, we can also deduce that the prediction of False Positives remains high, thus affecting the performance of our model. This unveils a drawback for our model and

with it, room for improvement. As the case may be and given the circumstances that we are dealing with user-generated data, the limitation exposed by our model indwells in the pre-processing stage. Where a deeper grammar correction a more strict named entity recognition strategy, can still be applied to increase the coverage of GloVe vocabulary. Because it shows the precision of non-spoiler and spoiler classification is almost equal to 0.70 and its AUC score is 0.702 which is the highest of the above-mentioned spoiler identification technique.

6.1 Conclusion

Our project to detect spoilers in movie reviews took us from a metadata analysis to text vectorization and pre-trained languages. Modeling from metadata showed us interesting genre information, but nothing practical in spoiler detection. Text vectorization showed much more promise although it'd cause many flagged posts which are spoiler-free. Using pre-trained models, we tapped into work done over billions of previous words and got better results. It was successfully built by combining the ability of a deep learning architecture like Bi-LSTM, altogether with a pre-trained GloVe model. The results obtained from our approach are considered to be satisfactory since our model proved that the usage of Neural Networks and pre-trained word embeddings, can equally maximize the discrimination capacity between classes. A total of six classifiers, including our proposal, were built and tested. It is consistent with the AUC score for every model, we've evaluated how each model performed. during this wise, we could demonstrate divergent levels of skill to tell apart between binary classes (True(1) or False (0)) that were delivered among the experiments

6.2 Future Work

However, there are some tasks that can be done to improve our model. Just like future work, that could focus on taking the pre-processing of text stage to the next level. We are still struggling to remove all this extra noise that comes in the form of grammar mistakes and name entities (movie names, brand names, character names) even after completing all the tasks already. Incorporating external services (e.g. Microsoft Language Understanding Intelligent Service, L.U.I.S) that are specifically intended to handle NLP tasks would help to increase coverage of the vocabularies for the co-occurrence matrix generated by the pre-trained GloVe method. The performance of the model can be further improved by hyper-parameter tuning, and also it might improve when we use any multilingual language models.

6.3 Code

Github Repository Link:`https://github.com/mowas455/Text_Mining_Project`

Bibliography

[1] Abdulaziz M. Alayba and Vasile Palade. Leveraging arabic sentiment classification using an enhanced cnn-lstm approach and effective arabic text preparation. *Journal of King Saud University - Computer and Information Sciences*, 2021.

[2] Karen Avetisyan1 and Tsolak Ghukasyan2. Word embeddings for the armenian language:intrinsic and extrinsic evaluation. *Evalution of word embedding techniques*, 2018.

[3] L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[4] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[5] Susana Eyheramendy, David Lewis, and David Madigan. On the naive bayes model for text categorization. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 12 2002.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press.

[7] Schmidhuber J Hochreiter S. Long short-term memory. *Neural Comput*, 1997 Nov.

[8] Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. Bag-of-embeddings for text classification. *wbow*, page 2824–2830, 2016.

[9] kanoki. Sentence similarity in python using doc2vec. https://kanoki.org/2019/03/07/sentence-similarity-in-python-using-doc2vec/.

[10] Gang Liu and Jiabao Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[11] Rishabh Misra. Imdb spoiler dataset, 05 2019.

[12] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. *Glove*, pages 1532–1543, October 2014.

[13] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian J. McAuley. Fine-grained spoiler detection from large-scale review corpora. *CoRR*, abs/1905.13416, 2019.

[14] Yabing Wang, Guimin Huang, Jun Li, Hui Li, Ya Zhou, and Hua Jiang. Refined global word embeddings based on sentiment concept for sentiment analysis. *IEEE Access*, PP:1–1, 03 2021.

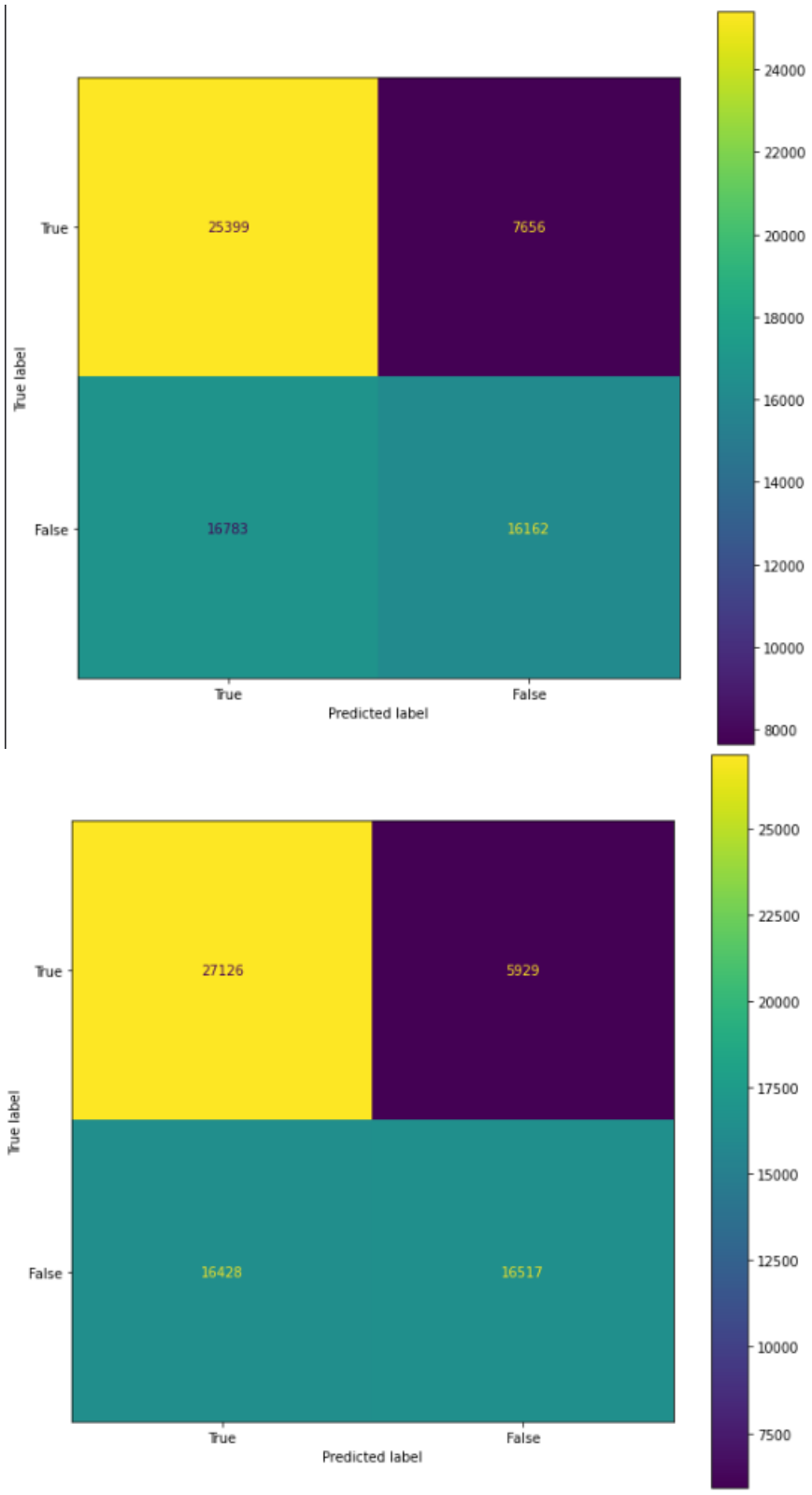

Figure 1: Confusion matrix for Naive Bayes and XGBoost Classifier

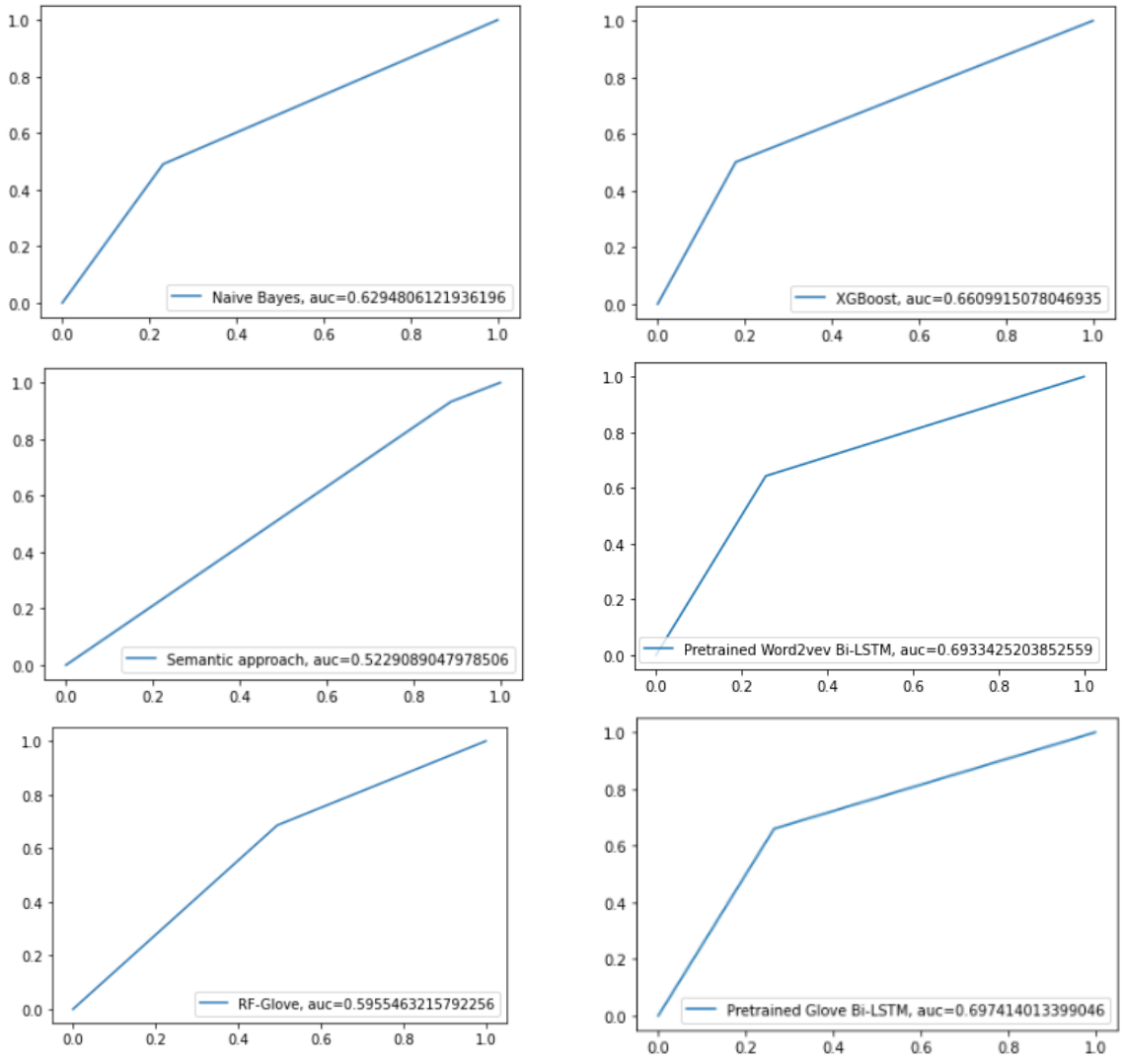

Figure 2: AUC scores for implemented models