Identification of Spoiler in IMDB movie review

Mowniesh Asokan (mowas455)

February 2022

Abstract

This paper presents the NLP (Natural Language Processing) approach to detecting spoilers in the IMDB review. Generally, these reviews reveal some information associated with the plot of a movie. An automated approach, filtering out such spoilers, would be ideal as manual labeling is impossible due to a large amount of content. To identify those reviews, we propose supervised machine learning models. So, we explored Bi-LSTM, XGBoost, Naive Bayes, and pre-trained GloVe to improve the accuracy in text classification. In addition to this, we used the Bag Of Words (BOW), cosine similarity, and Term-Frequency and Inverse Document Frequency (TF-IDF) method to process the text vectors. The results shown from our models are satisfactory. Quantitative and qualitative results demonstrate the proposed method substantially outperforms the baseline model.

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

Contents

Chapter 1

Introduction

Spoilers can destroy the experience of watching a movie. The final thing you wish when looking for movies to observe on IMDb and reading reviews to see how you wish to spend it slow is to come upon a spoiler that will ruin your experience. IMDb and similar sites do 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 are able to use movie details, descriptions, etc. But most significantly 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, except as spoilers instead of sentiments.

As spoiler annotations aren't universally supported, requiring manual effort, an automatic approach would be preferable. The task of detecting spoilers is nontrivial. as an 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 pre-trained language models and a few machine learning techniques, fine-tuning them using user-generated annotations to predict which documents contain spoilers. On the identical data, we'll also build a model that predicts the presence of spoilers on a token level, thereby marking sections of sentences as containing spoilers

1.1 Motivation

The aim and motivation of this project to build a novel supervised machine learning (ML) model with a natural language processing and pretrained Glove architecture to improve the accuracy of the text classification on 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 a trustworthy work that serves as foundation for future efforts against spoilers deal-breaker for global culture and film industry.

Chapter 2

Data

The data came from a Kaggle post by Rishabh Misra, which can also be found here. Overall, there were 573,913 reviews covering 1,572 different movies. Among these, 150,924 reviews were found to having 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 dataset to extract a useful information in movie details for predicting the spoilers. 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, there are some reviews contain a word spoiler.Therefore,it also may helps to find the spoiler text in the reviews.

We took the main feature for classification is review text and review summary.To work with the text, we need to change the text into consistent format by removing the stop

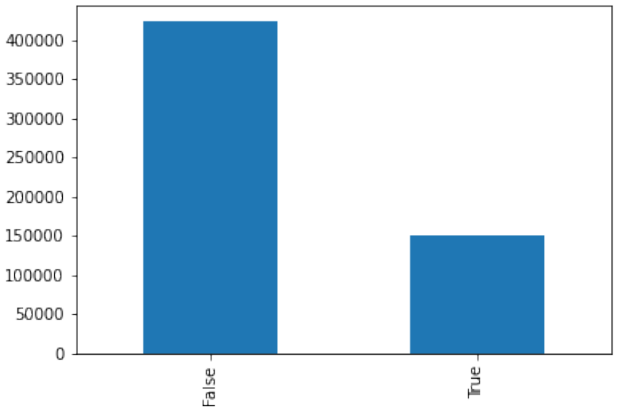

Figure 2.1: Complete distribution of spoiler data

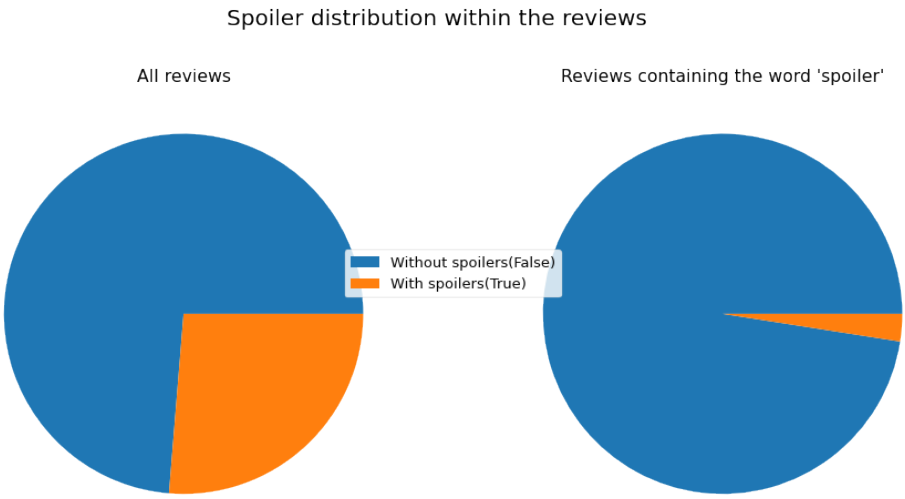

Figure 2.2: Sample movie spoilers in review

words, symbols such as question,exclamation marks in review text.This decrease size of the text and makes the model more effective to predict.During this stage normalization and transformation of text,such that tokenization of the text is one of the most efficient task of NLP activities, because help to identify the patterns and boundaries between every word in the text corpora.In the figure 2.1, the bar plot shows the complete distribution spoiler in the data-set.From the plot it clearly shows that distribution of the data is not equal.So we used 110000 observations which having a equal number of spoiler and non-spoilers

Therefore,all the pre-process actions will help the models to classify the review text as spoiler and non-spoiler.Since we are dealing with a large amount of text corpus with million of words,so there will be 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, and Glove. In addition to this neural network and some other methods, we will also discuss the Naive Bayes model, which will be our baseline. We discuss traditional metrics for evaluating machine learning systems in addition to more task-specific metrics that are less widely used.

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 of vanishing gradients. But, when we consider using long RNN's there is not much we could do with the traditional RNN's and hence it wasn't widely used. That is the reason that lead to the finding of LSTM's [7] which basically uses a slightly different neuron structure.In LSTM, we will be reffering to a neuron as 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 allow the network to have both backward and forward information about the sequence at every times 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 in the LSTM that 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

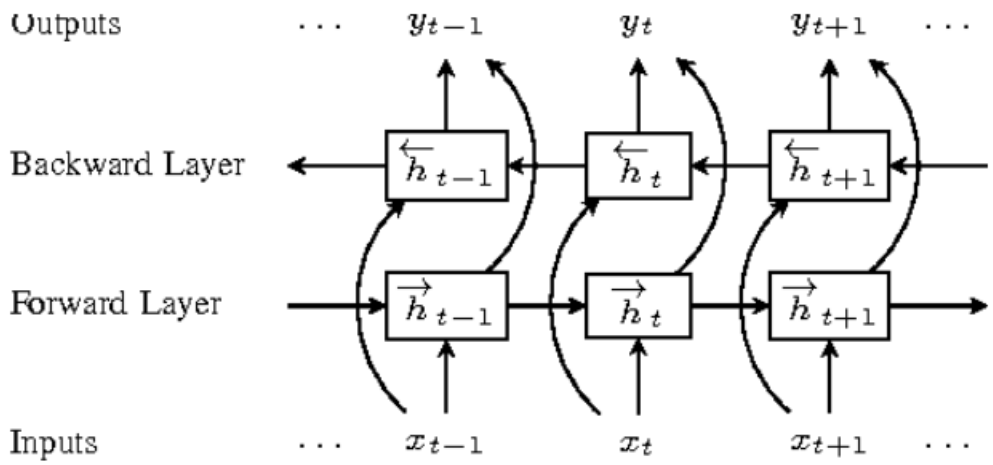

Figure 3.1: : BI-LSTM Architecture

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 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) 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 'maxdepth' 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 Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task and text categorization [5]. The crux of the classifier is based on the Bayes theorem.which is our **baseline model** for 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 values yes or no, for example if a spoiler in the text or not.The Bayesian equation and 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, Bidirectional Long Short-Term Memory Bi-LSTM with a pre-trained Glove and genism of word2vec embedding model[1]. For a machine learning model, 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.

As a result of word embedding, a set of characters, words, or sentences can be mapped to a set of continuous numbers that are used during the training phase of a computer system. we have chosen to utilize the following word representation techniques: Global Vectors (Glove), Term Frequency Inverse Document Frequency (TF-IDF), and Bag of Words (BOW). With the aim of representing 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 for Naive Bayes and XGBoost

Term frequency-inverse document frequency is a text vectorizer that transforms the text into a usable vector. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF).The term frequency represents the number of occurrences of the specific term in the document.Document frequency is the number of documents containing a specific term.TF-IDF is the just a multiplication of the term frequency and inverse documentfrequency.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 are 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 [13]. The embedding capture semantic meaning only when they are trained on a huge text corpus, using some word2vec model. Before training, the word embedding are randomly initialized and they don't make any sense at all. It's only when the model is trained,therefore that the word embedding have captured the semantic meaning of all the words.

4.2.1 Semantic based approach

Semantic based approaches deal with the relationship between the words. One of the best things 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[3]. 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 experiments uses the Semantic relations between word vectors and determine the similarity between two documents to solve spoiler identification problem.

4.2.2 BOW for Bi-LSTM

Word2vec is one of the popular technique to learn word embedding using a neural netork model.Word embedding via word2vec can make the natural language model to implement the mathematical operation on the words to detect the spoiler in the reviews[8].There are types of algorithms for word2vec - (CBOW) Continuous Bag of Word and Skip gram model.Genism word2vec requires that 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 bag of words for the word embedding.

Continuous Bag of Words(CBOW): In short, BOW is the most basic technique for representing sequences of documents. Because features are extracted by counting the number of words appearances in those documents. Dissimilar to GloVe and TF-IDF models, BOW does not really bargain for words organization nor grammar. And it is said to be a good technique when used over small data. BOW features will be use alongside with Bi-lstm model.

$$Bow_3 = Bow_1 + Bow_2 \tag{4.3}$$

4.3 Pre- trained Glove embedding for Bi-LSTM model

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.

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_{ik}}{P_{jk}} \tag{4.4}$$

Specifically, GloVe [11] 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 actually 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 in order to achieve higher coverage values.

4.4 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.

$$TPR = \frac{TP}{TP + FN} \tag{4.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.6}$$

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

Chapter 5

Implementation and Result

In our case, we conducted the 5 cases of study comprising the three different technique,therefore model performance, results with TF-IDF vectorizer, Genism and Pretrained-GloVe are shown below.

5.1 Naive Bayes with TF-IDF

This is the first experiment of spoiler identification and we fix this as baseline model. we have chosen to train by Bernoulli Naive Bayes Classifier and Using small amounts of data in the training stage, we aim to increase the Recall value per class while maintaining a fast computational ability. The Bernoulli NB algorithm was specifically chosen for this experiment because of the distinct penalty for non-occurrence, which this type of NB algorithm applies over binary features.

5.2 XGBoost with TF-IDF

In this case, we attempted to built a good classifier for our text corpora[12], by appealing the potential of eXtreme Gradient Boosting Algorithm.While fitting this data into a machine learning model, it requires the data in the more specific state (transformed form) to classify the spoilers. we have chosen the parameter eta is 2 and objective function is binary: *logitraw*.Moreover the resulting test accuracy and f1 score is relatively higher than the Naive Bayes model.The Area under the curve (AUC = 0.66) is also have some improvement.

5.3 Semantic approach with Review and Plot summary

Word embedding is build on the concept of Continuous bag of words and one hot vector. This made Word2Vec most efficient in terms of determining the similarity by calculating cosine distance [2]. Similarity is determined by comparing word vectors or word embedding, multi-dimensional meaning representations of a word. Word vectors can be generated using an algorithm like word2vec. In this approach, a vector of every word in each document is considered as whole and cosine similarity [9] is calculated between them. A threshold value is

used a hyper parameter and the classification of reviews is done based on it. When threshold value = 0.9, this approach has shown very low result by classifying the data

5.4 Bi-LSTM with BOW

We tried the Bag of Words (BoW) model next to predict spoilers. Bag of Words is a method for converting text into a vector which we can run gensim-word2vec model. First, we build a corpus by parsing each review and adding every unique word. we additionally checked the phrases for the review text with bigrams and trigrams function. In the end, pre-processed texts are converted into the sentence to sentence with ngrams and finally those sentences are tranformed into the padded sequences for Bi-lstm model.The time taken for processing the sentences to ngrams is very high.So, we used only 40000 observation for this methods.

5.5 Bi-LSTM with GloVe

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

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	0.74	0.50	0.60	0.66	0.66
Bi-LSTM- BOW	non-spoiler	0.65	0.69	0.67		
	spoiler	0.68	0.63	0.66	0.66	0.66
Semantic similarity	non-spoiler	0.11	0.63	0.19		
	spoiler	0.93	0.51	0.66	0.52	0.52
Bi-LSTM- GloVe	non-spoiler	0.69	0.73	0.71		
	spoiler	0.71	0.67	0.69	0.70	0.699

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 a 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 the table 5.1, the Naive Bayes classifier achieved the fair precision score by correctly classifying the class spoiler with a score of 0.68. However, the classifier did not perform as good over 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 a XGBoost model.It shows the precision result of spoiler is improved to 0.74 and non-spoiler is to be 0.62,on account of this AUC score of this model also improved compare with NB-Classifier.These lack of prediction is evidentially proven in confusion matrix result of both the model.

To improve the prediction result, we moved to the BOW technique with help of bigrams, trigrams and word2vec function.Therefore the padded sequence of bag of words is trained into the Bi-LSTM model to get prediction. This method shows the **significant change** in the result of spoiler and non-spoiler precision score of 0.65 and resultant AUC score is also 0.65. Therefore the BOW with Bi-LSTM model learned the Classification task equally.By the help of same BOW 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 is *spoiler*, because review text revealing the original plot summary of a movie.On the other hand, if it less then it is *non-spoiler*. Even though it is conceptually gives meaning, but it does not work fine with prediction score that shows in table 5.1.

On the other side of the coin our proposed model for classification of movie reviews, achieved the highest AUC score out of all five models.Our Bi-LSTM model using pre-trained GloVe technique, can predict high numbers of True Positives and a fairly number of True Negatives. Although, we can also deduce that the prediction of False Positives remains high, thus affecting the performance of our model. Evidently, 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 an more strict named entity recognition strategy, can still be applied in order 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 it's AUC score is 0.702 which is 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. Modelling 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 combining the ability of a deep learning architecture like Bi-LSTM, altogether with 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 pretrained word embeddings, can equally maximize the discrimination capacity between classes.A total of 5 classifiers, including our proposal, were built and tested. 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 still tasks that can be done in order 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, etc.); 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 also be improved by taking into account different languages during pre-processing.

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] analytic vidya. From count vectors to word2vec:. https://www.analyticsvidhya.com/blog/2017/06/wordembeddings-count-word2veec/.

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

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

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

[13] 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.