Spoiler Detection from Movie Reviews

Prathamesh Pandit
pppandi2@ncsu.edu
NC State University
Raleigh 27606

Rohit Nair
rnair2@ncsu.edu
NC State University
Raleigh 27606

Sanveg Rane
ssrane2@ncsu.edu
NC State University
Raleigh 27606

Saurabh Mhatre
smhatre@ncsu.edu
NC State University
Raleigh 27606

ABSTRACT

It is not uncommon for people to base a large portion of their decisions of watching or not watching movies on online reviews. In most cases, these reviews complement the movie genre in that they give the users a deep enough dive into the movie for them to decide if it is something that interests them. However, like everything else, there are advantages and disadvantages of online movie reviews. A big disadvantage which only seems to be getting worse with every passing day is that of spoilers in movie reviews. Many times, people often end up revealing the plot or a part of the plot in reviews, thereby robbing the reader of the experience of enjoying the movie. In this paper, we discuss a potential solution to address this problem. We plan to use a combination of various Machine Learning techniques to analyse online reviews in an attempt to detect the ones containing spoilers. Additionally, we discuss the approaches we followed and issues we encountered leading up to the final model.

KEYWORDS

dataset, sentence similarity, semantic similarity, artificial neural networks, recurrent neural networks, machine learning, natural language processing, accuracy, precision, long short-term memory, natural language inference, enhanced sequential inference model

1 BACKGROUND AND INTRODUCTION

1.1 Problem Statement

Spoilers are everywhere on the Internet. With no real warning or indication of spoilers in reviews, a lot of people often end up reading such reviews which keeps them from experiencing the movie in all its glory. Also, there do not seem to be a lot of easily accessible public tools to prevent this issue. We aspire to address this issue by applying various Natural Language Processing techniques to the IMBD Spoiler Dataset which is hosted on Kaggle. The dataset consists of two parts. The first one contains data about movies and the second one contains review data. We intend to extract meaning from both - the movie dataset and the review dataset after performing necessary pre-processing in an attempt to find similarities between the two and classify a review as containing spoiler or not.

We implemented a technique for learning sentence similarity provided by Mueller et. al. [4] which proposes the Manhattan LSTM model. In our implementation we provide word embedding vectors of plot information and review to two LSTMs for comparison, to generate fixed size vectors that encode meaning expressed in them. We later compare these vectors to understand how close the review is to plot information.

We also explored some techniques in Natural Language Inference, and decided to implement the Enhanced Sequential Inference Model[1]. In NLI we try finding logical, semantic relationship between premise and hypotheses as either entailment, neutral or contradiction. For spoiler detection, this approach can be used where we consider plot summary as the premise, movie review as hypotheses. However this problem contains only entailment as spoilers and neutral for non-spoilers.

1.2 Related Work

LSTM [3] networks have over time been proven to be powerful language modelling techniques, capable of language tasks requiring intricate understanding. Sutskever, Vinyals, and Le (2014) [5] demonstrated that an effectively trained network maps each sentence onto a fixed-length vector that reasonably encodes the underlying meaning expressed in the text. Other LSTM variants like the simple gated recurrent unit (GRU) of Cho et al. (2014) [2] were also proposed, however their effectiveness and reliability over LSTMs cannot be concluded as overall better or worse compared to LSTMs. A technique for learning sentence similarity was provided by Mueller et. al. [4] which proposes the Manhattan LSTM model. MaLSTM is simply a model that uses two LSTMs to measure similarity between a pair of sequences (sentence paragraphs in our case). We pass sentence vectors that are generated with the help of pre-trained Word2Vec vector embedding hosted by Google. The two LSTMs convert the variable length sequence into a fixed dimensional vector embeddings and then they are compared using a similarity measure like Manhattan distance to find if they are similar in meaning. A different technique for obtaining inference from Natural Language is demonstrated using the ESIM (Enhanced Sequential Inference Model) model [1] provided by Chen et. al. They present a technique which makes use of various stages of processing like Encoding the input, Inference Modelling, Inference Composition and Pooling to obtain predictions whether the inference generated from both sentences are similar or not.

2 METHOD

2.1 Approach

Machine learning models are functions or algorithms which are used to predict some output for a particular input based on previous patterns of some input. Upon exploring different RNN techniques, we decided to proceed with using Long Short-Term Memory (LSTM) network for creating our model and build upon it by performing various enhancements. In the initial phase, we implemented a Manhattan Long Short-Term Memory network model. MaLSTMs are based on artificial recurrent neural network architecture and have several advantages over conventional neural networks. In addition to using this as a starting point to progress toward the final model, we also used the results of this model in combination with other techniques for feature selection during the pre-processing phase. Further, in an attempt to improve the accuracy and get better predictions, we came across a Natural Languages Inference model ESIM, and decided to implement the same.

2.1.1 Long Short-Term Memory (LSTM). Long Short Term Memory networks or "LSTMs" are special kind of RNNs, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. They have the ability to remember / persist information across long periods of time which is crucial for sequential tasks like Natural Language Processing and Inference.

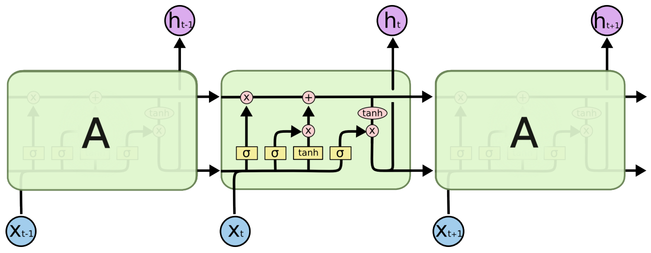


Figure 1: LSTM Module Architecture

All recurrent neural networks have the form of a chain of repeating modules of neural network layers. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs have a chain-like structure with repeating modules or cells of neural network layers. Each module contains four different network layers that have a specific interaction between them that generates an output for the current input and also generated input for the next module in the sequence.

Following equations explain the computations performed inside every cell.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C_t} = tanh(W_C.[h_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_i * \tilde{C_t} \tag{4}$$

$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

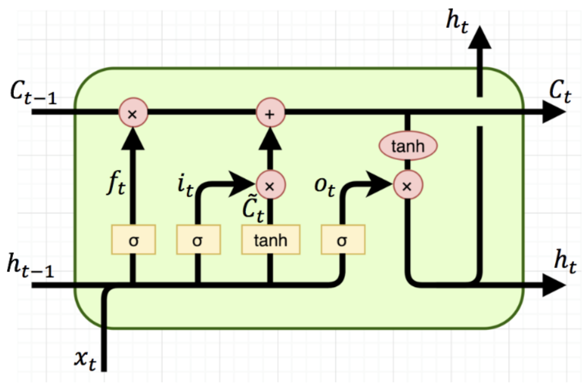


Figure 2: LSTM Cell

2.1.2 Manhattan LSTM (MaLSTM). We initially decided to perform the task of Learning Sentence Similarity using Manhattan LSTM[4] (MaLSTM) Model described by Mueller et. al. MaLSTM is simply a model that uses two LSTMs to measure similarity between a pair of sequences (sentence paragraphs in our case). We pass sentence vectors that are generated with the help of pre-trained Word2Vec vector embedding hosted by Google. The two LSTMs convert the variable length sequence into a fixed dimensional vector embeddings and then they are compared using a similarity measure like Manhattan distance to find if they are similar in meaning.

2.1.3 Enhanced Sequential Inference Model (ESIM). ESIM [1] uses various stages of processing to generate the relationship between premise (movie synopsis) and hypothesis (movie review) as entailment (is a spoiler). This model uses several layers of processing to extract semantic meaning between two encodings and tries to infer semantic similarity between them. Initially, sentence embeddings are generated and encoded using Bi-LSTM. Later on, Soft Alignment is done with the computation of attention weights. This is followed by concatenation and then composition using Bi-LSTM to help provide better semantic intuition. This data is then passed to a few dense layers, along with Pooling and finally, results are generated using Softmax.

2.2 Rationale

Neural Networks are machine learning techniques that use a combination of input and output layer coupled with one or more hidden layers which are connected by weights. These weights are adjusted (i.e., learning takes place) to improve the accuracy. However, in case of traditional Neural Networks the inputs and outputs are independent of each other. This exposes a major hurdle associated with neural networks - these cannot be used in case of sequential data like text or time series where the current state is dependent on the previous state. This is where Recurrent Neural Networks come into the picture. RNNs have a sense of memory which allows them to gain some context. LSTMs further build upon RNNs and create a short-term as well as a long-term memory component.

To detect if a review contains a spoiler or not, the context in which a particular word is used plays a crucial role. We need to be able to understand the contextual meaning of the various words which comprise a review and the corresponding movie plot to be able to tell if both the texts mean the same thing. We feel LSTMs will help us achieve just that.

3 EXPERIMENT

3.1 Dataset

We will be using the IMBD Spoiler Dataset which is available on Kaggle. The dataset consists of information about movies including summaries and/or synopsis in addition to other metadata about the movies. It also contains information about reviews corresponding to the movies along with a label to indicate if the corresponding review contains a spoiler or not. The movies dataset contains more than 1500 records whereas the reviews dataset contains more than 500K records.

The movie details contains metadata for the movies with attributes movie_id, plot_summary, plot_synopsis, duration, genre, etc. Of these, the plot_summary attribute contains a summary of the movie's content and doesn't include spoiler content. The attributes plot_synopsis is of importance as it is a synopsis of the movie's content and contains spoiler content

The reviews contains user reviews for various movies consisting of attributes movie_id, review_date, user_id, review_text, rating, is_spoiler. The attributes review_text and is_spoiler are of concern in our use case. The review_text is used to compare with the plot_synopsis and the corresponding is_spoiler indicates whether the given review contains spoiler.

3.2 Hypotheses

The experiment aims to recognize whether the given review contains any spoiler information which could impair a reader's movie-watching experience. The main source of information and spoilers for the movie would come from the summary and synopsis, thus re-framing the purpose of the experiments conducted to match the content of movie synopsis and movie summary to a user's review to detect more matching content.

Determining a logical relationship between the movie details, which are known to contain spoiler data, and the user review using Natural Language Inference could help to understand the degree of spoiler content in the user review. The relationship determined to contain entailment, which indicates a presence of spoiler, or neutral which in indicates no spoiler in the review.

3.3 Experimental Design

To estimate the results of the stated hypothesis, we implemented Recurrent Neural Network (RNN) models to determine the relationship between movie synopsis, which contains the spoiler data and the user movie review to determine whether the review contains any content that can be considered a spoiler. Two RNN models are implemented, the first using Traditional Manhattan distance on LSTM encodings, and later using Enhanced Sequence Inference techniques. Both models take the same pre-processed data in the form of word2vec embeddings as input to determine the logical relationships amongst texts.

(1) **Data Pre-processing:**
Data pre-processing is an important step which helps to standardize the data in a format such that it can be used by the model for training. As a part of this activity, we performed the following steps -

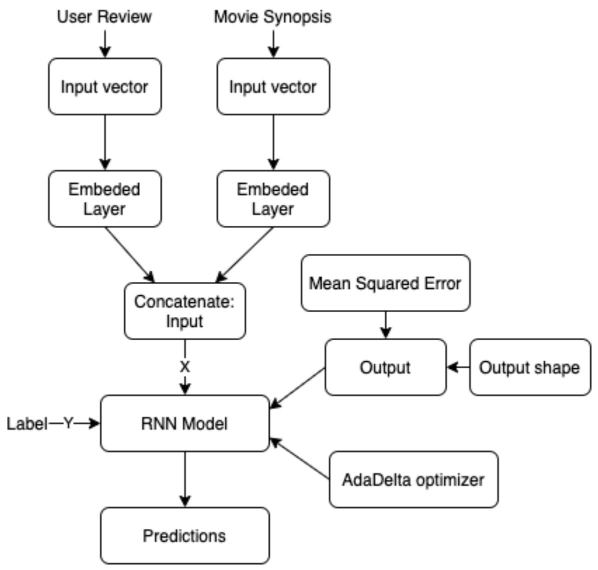


Figure 3: Experimental design setup

- Remove unwanted columns - We removed columns from the dataset which did not directly contribute to our analysis. Since our text analysis involved only a few attributes like Plot Summary, Plot Synopsis, Review Title and Review Content, we ended up not using any of the other attributes.
- Removing missing values - A major requirement of the analysis was the presence of content in the attributes we had selected, as mentioned earlier. We removed records for which these attributes were missing.
- Stratified Sampling - The dataset contains a total of 500,000 of which the experiment uses a total of 300,000 rows. To evenly distribute samples of spoiler and non-spoiler reviews, stratified sampling strategy is applied. Of the total dataset, about 150,000 reviews are classified as containing spoiler texts. The dataset is sorted by the label field and extracted such that the 300,000 rows contain a mix of equally distributed spoiler and non-spoiler user reviews.

Thus our data pre-processing phase was mainly centred around removing attributes unwanted attributed from the dataset and associating reviews from the review dataset with corresponding movies from the movie dataset. Additionally, the model we plan to implement is majorly concerned with a few attributes - the review text, the review label and the corresponding movie synopsis and/or summary.
We planned to make use of the movie synopsis and/or summary attribute in combination with the review text to find similarities among the two and predict if the review contains a spoiler or not. To decide which attribute among the movie synopsis and movie summary corresponds more closely with the review text, we decided to create three sets of training data which differed in the one attribute - the movie summary

and/synopsis. The first set contained the movie summary attribute, the second contained the movie synopsis attribute and the third contained the one which had the longer length from among the two.

(2) **Input generation:**
The user review and reference data together form the input for the LSTM model. Reference data is used to compare the review text to detect the presence of spoiler content and is either only the Movie Synopsis, only Movie Summary or a hybrid created using both the fields. Detecting which reference data provides accurate training of the model also forms a part of the experiments. Reviews similar more similar to reference data generally identify as spoilers and thus, the reference data must be researched to find better representations.
The input fields are converted to a vector for similarity calculation. Each word in the entire dataset is assigned a unique id to represent it as a vector. The entire strings are then converted to vectors using the assigned id. This is used for computing the similarity of texts during the training of the Manhattan LSTM model.

(3) **Manhattan LSTM Model:**
- Manhattan distance calculation
  The similarity of the texts, converted to the vectors, is calculated using the Manhattan distance. The initial step is calculating the distance vector by taking the absolute difference between the two vectors of input texts. Further, negative exponent of the difference vector which intuitively if closer to 1 indicates the two compared vectors to be similar. A lower value of the exponent indicates more dissimilarity amongst the text vectors.
- Model Training The model computes a similarity of between two texts and compares the similarity using the negative exponent Manhattan distance calculation. The calculated distance is then compared to the label field in the training data which indicates if the review was accurately classified as a spoiler.
  The LSTM model uses a 50-dimension representation of nodes in the hidden layers. The Ada-delta optimizer is used for the optimization of the parameters. The learning rate for the LSTM network is set to 0.1 with 10 epochs. Of the 250,000 rows, 200,000 rows are randomly selected for training and remaining 50,000 for testing the trained model in each epoch. The model works by calculating the similarity of the two sentences and for each instance in the training set, the review text is compared to the movie's reference data selected as per the model.

(4) **Enhanced Sequential Inference Model:**
The Enhanced Sequential Inference Model (ESIM) aims to infer semantic similarity between two encoded vectors of text by utilizing several layers of processing. The model uses Bi-LSTM for Natural Language Inference which forms the basis to generate a logical relationship between text vectors to compare the similarity between a user review on a movie and the corresponding movie synopsis.

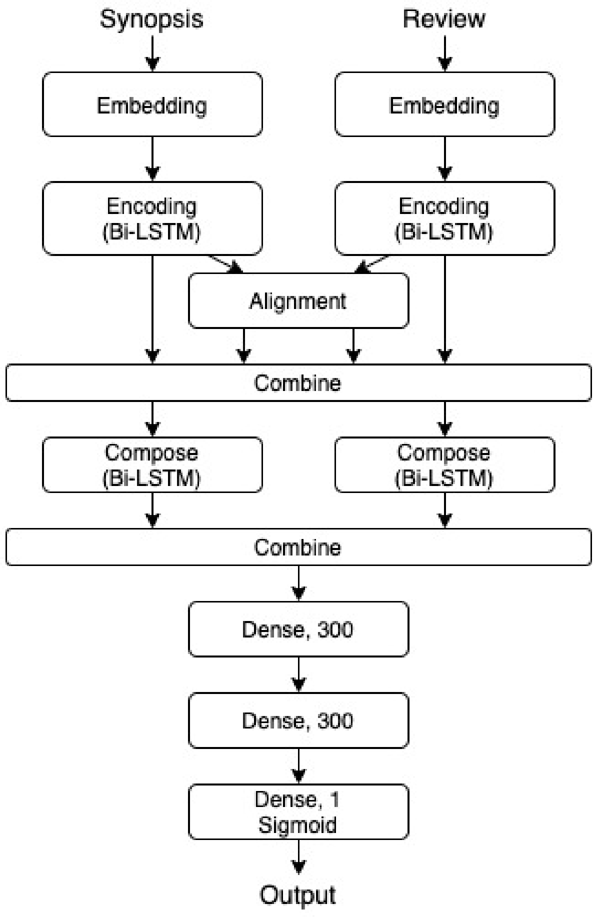


Figure 4: ESIM Flow

- Input Encoding Layer
  The embedded input vectors are converted to encoding using a Bi-LSTM layer which is the initial step to NLI. The first Bi-LSTM layer is used to learn a word from the vectorized input and learn its context. This layer starts a bi-directional LSTM on a sequence starting from left and right ends, and the hidden layer weights generated by there LSTMs at each step represents the time step of the word and its context.
- Local Inference Modeling
  The local inference model forms the basis of inference of relationship between the hypothesis and premises. The encoded inputs from the Bi-LSTM layer are processed through the alignment layer and then concatenated as input to the Composition layer. This layer computes the similarity of a hidden state tuple between the input statements. These hidden weights are used to compute local

inferencing which provides the relevance between the hypothesis and premises. The local inference is enhanced by calculating the difference and pairwise product for each tuple in the encoded inputs

- Composition Layer
  The second Bi-LSTM layer performs inference composition where the local inference data (the alignment) and its interaction is combined to form a single output vector. This layer aims to capture the local information and the context for inference composition.

- Model Training
  The final ESIM model is a combination of the Input Encoding layer, the Inference Modeling Layer and Composition layer, the outputs of which are fed to a Dense layer of 300 neurons and finally passed to a single neuron Dense layer with a 'Sigmoid' activation function. The output of the final layer is a probability value, with a higher value indicating the presence of spoiler content in the review and vice versa.
  The model is a combination of layers which take in the premise and hypothesis as vectors and compute their similarity my making natural language inference which is further inputted to a neural network to predict the probability of the relationship between them.
  The final neural network is a combination of three dense layers, input to which is a set of tuples comparing the similarity of the Review text and the Synopsis text. The two dense layers contain a total of 300 neurons which input the values to the final layer with a single neuron and a sigmoid activation function. The model is trained on a data set of 230,400 rows and validated on 57,600 rows for learning in each epoch.

4 RESULTS

4.1 MaLSTM results

The Manhattan LSTM model was trained by providing Movie Synopsis and Movie Review pairs. Fig 5 and Fig 6 represents the Model performance with respect to Accuracy and Loss metrics respectively. This model was trained using Adam Optimizer (lr=0.001) with a batch size of 64 for 16 epochs. We implemented early stopping and were able to obtain a test accuracy of 0.75 and a test loss of 0.17. Fig 7 represents the Precision, Recall and F-1 Score of the model.

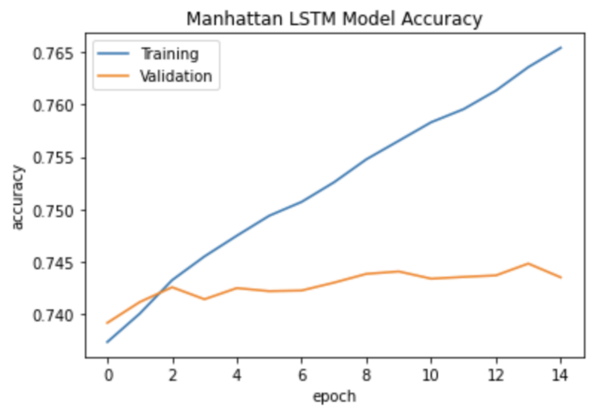


Figure 5: Manhattan LSTM Model Accuracy

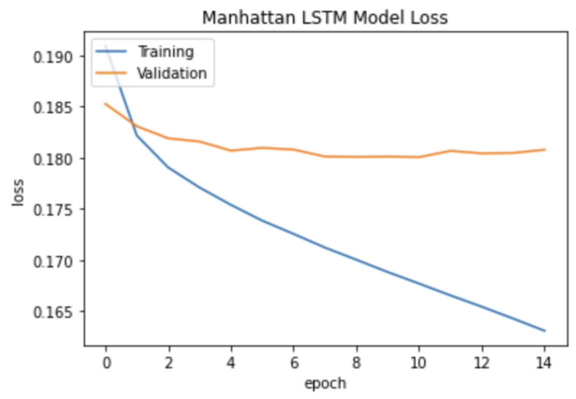
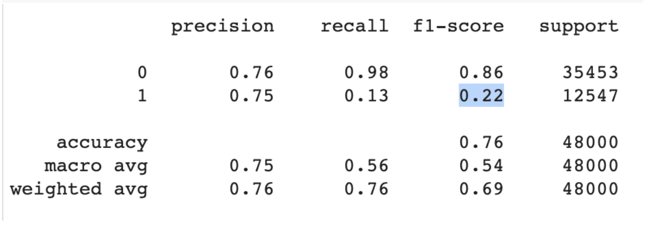


Figure 6: Manhattan LSTM Model Loss


```
              precision    recall  f1-score   support

           0       0.76      0.98      0.86     35453
           1       0.75      0.13      0.22     12547

    accuracy                           0.76     48000
   macro avg       0.75      0.56      0.54     48000
weighted avg       0.76      0.76      0.69     48000
```

Figure 7: Manhattan LSTM Model Performance metrics

4.2 ESIM results

The ESIM model was trained on the same data. Fig 8 and Fig 9 represents the Model performance with respect to Accuracy and Loss metrics respectively. This model was trained using Adam Optimizer (lr=0.0003) with a batch size of 64 for 16 epochs. We implemented early stopping and were able to obtain a test accuracy of 0.77 and a test loss of 0.16. Fig 10 represents the Precision, Recall and F1 Score of the model.

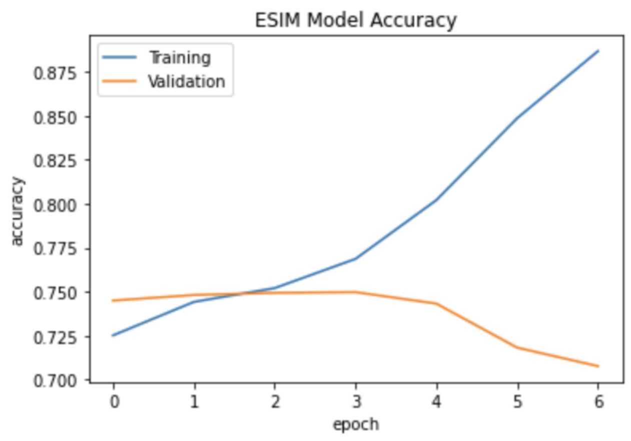


Figure 8: Enhanced Sequential Inference Accuracy

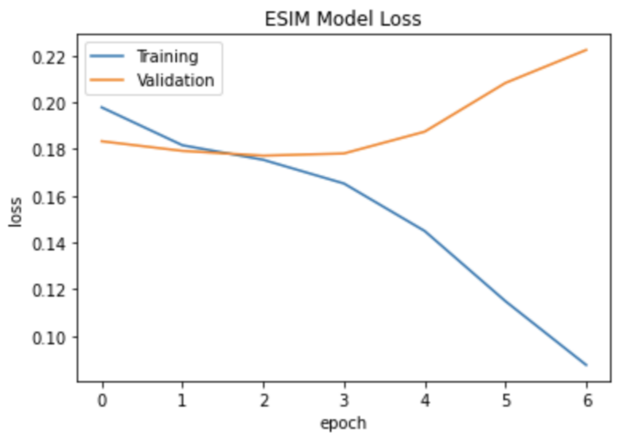
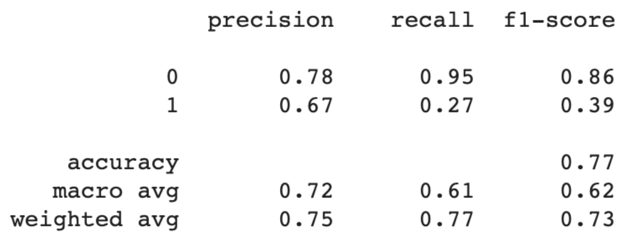


Figure 9: Enhanced Sequential Inference Loss

4.3 MaLSTM results using stratified data

The MaLSTM model was trained on data with stratified sampling. Fig 11 and Fig 13 represents the Model performance with respect to Accuracy and Loss metrics respectively. This model was trained using Adam Optimizer (lr=0.001) with a batch size of 64 for 16 epochs. We implemented early stopping and were able to obtain a test accuracy of 0.61 and a test loss of 0.23. Fig 12 represents the Precision, Recall and F1 Score of the model.



```
              precision    recall  f1-score

           0       0.78      0.95      0.86
           1       0.67      0.27      0.39

    accuracy                           0.77
   macro avg       0.72      0.61      0.62
weighted avg       0.75      0.77      0.73
```

Figure 10: Enhanced Sequential Inference Performance metrics

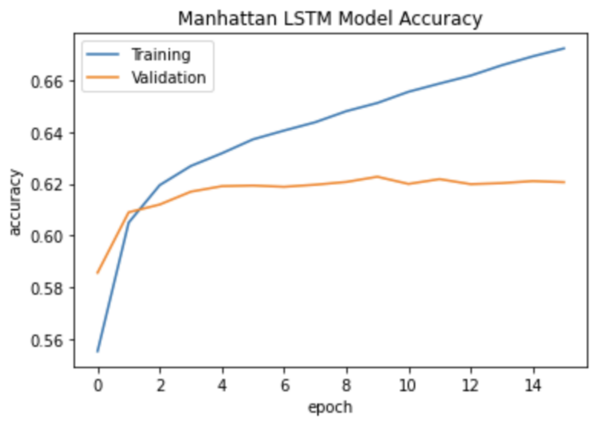


Figure 11: MaLSTM Accuracy with stratified sampling

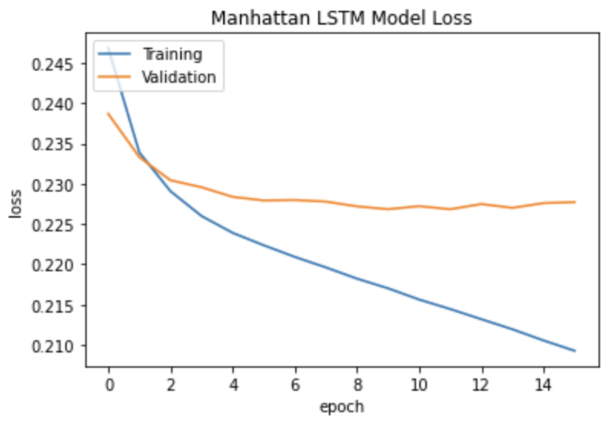
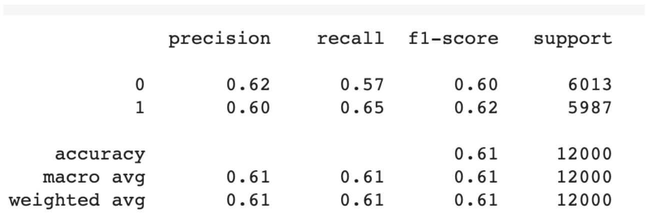


Figure 12: MaLSTM Loss with stratified sampling

4.4 ESIM results using stratified data and class weights

The ESIM model was trained on stratified data, however class weights were defined for training the model. Fig 14 and Fig 15 represents the Model performance with respect to Accuracy and Loss metrics respectively. This model was trained using Adam Optimizer (lr=0.0003) with a batch size of 64 for 16 epochs. We



```
              precision    recall  f1-score   support

           0       0.62      0.57      0.60      6013
           1       0.60      0.65      0.62      5987

    accuracy                           0.61     12000
   macro avg       0.61      0.61      0.61     12000
weighted avg       0.61      0.61      0.61     12000
```

Figure 13: Performance metrics for MaLSTM with stratified samplings

implemented early stopping and were able to obtain a test accuracy of 0.62 and a test loss of 0.22. Fig 10 represents the Precision, Recall and F1 Score of the model.

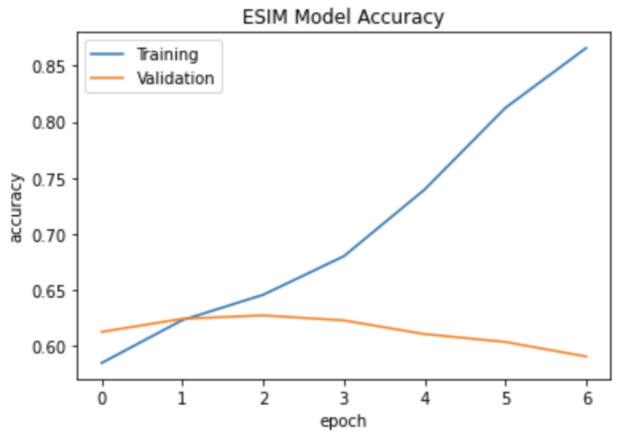


Figure 14: Enhanced Sequential Inference Accuracy with stratified sampling

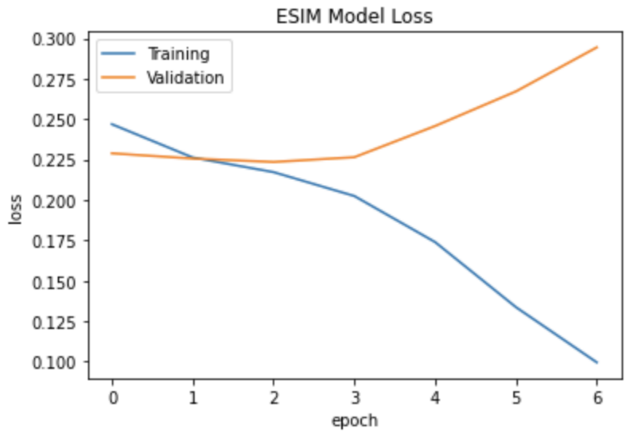
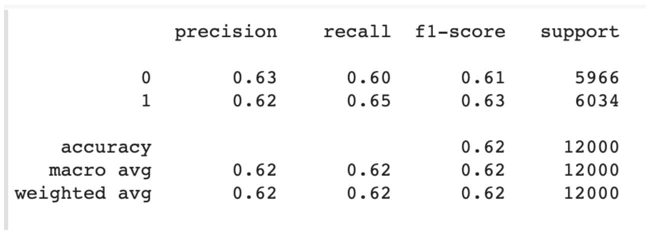


Figure 15: Enhanced Sequential Inference model Loss with stratified sampling



```
              precision    recall  f1-score   support

           0       0.63      0.60      0.61      5966
           1       0.62      0.65      0.63      6034

    accuracy                           0.62     12000
   macro avg       0.62      0.62      0.62     12000
weighted avg       0.62      0.62      0.62     12000
```

Figure 16: Performance metrics for ESIM with stratified sampling

4.5 Insights and Model Comparison

Upon comparing the different metrics provided by both the models, we can observe that the ESIM model provides higher test accuracy and lower test loss as compared to MaLSTM model. ESIM model also has a better F-1 Score, which can be considered as a better evaluation metric than accuracy for our problem. Therefore we conclude that the ESIM Model is better than MaLSTM for Spoiler Detection. Looking at the low values of Recall in both models, we realized that we needed to emphasize more on training the model with samples that contained spoilers. Therefore we decided to perform stratified sampling along with providing class weights emphasizing more on examples with target class denoting spoiler. We were, therefore, able to improve the Recall and F-1 Score Values significantly.

5 CONCLUSION

Analyzing the performance metrics of the Manhattan LSTM and ESIM models, and comparing the Accuracy, Loss and F-1 Score for our classifications, we can conclude that the overall performance of the model in detecting the presence of Spoilers in the user review is influenced by the technique used for generating and comparing the semantic representation of the texts.

The Manhattan LSTM model compared the similarity of words in the texts and made classifications based on the similarity of the vector encoded User review and corresponding Movie Synopsis. The approach is largely based on the number of words present in each text files and influenced by the size of texts. The model falls short in scenarios where data is sampled randomly and the user review contains a lot of text but not a lot of content that can be considered as spoilers. This causes higher dissimilarity between the synopsis and the review content causing the model to classify it as not a spoiler (i.e. False class), but the presence of even a small amount of spoiler content makes the review a spoiler.

The Enhanced Sequential Inference model attempts to establish a logical relationship between the review and synopsis texts by using a Natural Language Inference approach. The model first analyzes the input texts, develops a local inference of the relationship of the texts and converts the input to a vector data indicating the context of the review per the spoiler containing synopsis. This technique improves the results of predictions made by the model as it is has a better comparison of the texts.

6 GITHUB REPOSITORY

https://github.ncsu.edu/pppandi2/CSC522-P24-Spoiler-Detection-in-movie-Reviews

7 MEETING ATTENDENCE

	Date	Time	Attendees
1	April 10^{th}	4:00 - 6:30 PM	Prathamesh Pandit, Rohit Nair, Sanveg Rane, Saurabh Mhatre
2	April 17^{th}	4:00 - 7:00 PM	Prathamesh Pandit, Rohit Nair, Sanveg Rane, Saurabh Mhatre
3	April 21^{st}	6:00 - 8:30 PM	Prathamesh Pandit, Rohit Nair, Sanveg Rane, Saurabh Mhatre
4	April 23^{rd}	6:00 - 9:30 PM	Prathamesh Pandit, Rohit Nair, Sanveg Rane, Saurabh Mhatre
5	April 24^{th}	4:00 - 10:30 PM	Prathamesh Pandit, Rohit Nair, Sanveg Rane, Saurabh Mhatre

REFERENCES

[1] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017). https://doi.org/10.18653/v1/p17-1152

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. (2014). arXiv:cs.NE/1412.3555

[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[4] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. (2016), 2786–2792.

[5] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. arXiv:cs.CL/1409.3215v3