GENRE CLASSIFICATION AND SENTIMENT ANALYSIS OF STEAM GAME REVIEWS

CSCI 4152 – NATURAL LANGUAGE PROCESSING

ASAD KOTHAWALA (P-22) - B00825551

15TH DECEMBER 2021

ABSTRACT

In this paper I discuss the benefits of pre-trained word embeddings for sentiment analysis and multi-label classification tasks on video game reviews. The project uses reviews from Steam Games Library and hopes to highlight techniques that can help developers, distributors and customers get the most from the feedback provided through the store and online forums.

1. INTRODUCTION

Sentiment Analysis or opinion mining is a natural language processing technique that is used to determine whether given text is positive or negative – sometimes a neutral consideration is also taken into account. This process is of great help to businesses that monitor a brand or need to act on product sentiments through customer feedback. In many industries understanding customer needs is vital to smooth operation and a growing business. In this project I hope to highlight several natural language processing techniques as well as the usefulness of pretrained word embeddings.

Similarly, genre classification using a multi-label architecture can provide businesses that distribute multi-genre content feedback on customer preferences by extracting information on the most talked about genres in the media. To this I've have proposed a few multi-label classification models that can classify user reviews to predict the genre of the game.

I am also interested in whether using a pre-trained word embedding that are trained over similar kind of text language – I use GloVe trained over tweets that are also expected to have a lot of modern language and terms – will help the performance of my models for this task.

2. RELATED WORK

As part of my research for this project I found a number of papers that worked on similar problems. As mentioned by Jiang and Zheng [1], genre classification of video games is extremely challenging because of the number of genres a game can have, sarcasm in the reviews and the use of modern 'lingo'.

Sentiment Analysis is also similarly affected. Viggiato et al. [2] explore the problems with classifying sentiment of game revies. They found that most classifiers did not perform well. One of the causes they identified were the presence of reviews that point out both the advantages and the disadvantages of the game.

Zhen Zuo performed Sentiment Analysis of Steam Reviews using Naïve Bayes and Decision Tree Classifiers. They found that their best accuracy was approximately 75% using the decision tree classifier.

In their paper, Jiang and Zheng used a multi-modal approach to classifying genre's, however, their pure text model had top 1 accuracy of only 47.%. In their sentiment classification approach, Viggiato et al. had achieved an F1-score of 0.65 on their best performing model.

3. PROBLEM STATEMENT

The Video Game Industry has been growing at a rapid pace since the 1980's. By 2025 it is estimated to be worth $256.97 billion. One of the largest video game distributors today, "Steam", is an entirely online

store that allows customers to access their listings and download games through the web. Users also make use of the same platform to rate, review and leave feedback for the games they buy. This has resulted in a highly populated dataspace that is currently being underutilized by the industry as it is very difficult to manually process so many reviews and messages. In this project I hope to explore methods by which the customer, developer and distributor of video games may benefit from the use of Natural Language Processing techniques in order to make more informed decisions. To do this, I have compared the performance of several word embedding techniques on two important problems. The first is Sentiment Analysis of the reviews and the second is Multi-Label Genre Classification. Through sentiment analysis I believe customers and developers of games can have an improved understanding of the public response of a game. Genre classification can help distributors of games make decisions on what new games to greenlight for their store as it will help them understand their market better.

4. METHODS

A. DATASET

Video Game Reviews from Steam were retrieved from a Kaggle dataset containing over 6.4 million reviews. The dataset contained only five columns as shown in Figure 1 [3].

app_id Game ID	app_name Game Name	review_text Review text	review_score Review Sentiment: whether the game the review recommends the game or not.	review_votes Review vote: whether the review was recommended by another user or not.
10 565k	[null] 3% PAYDAY 2 1% Other (6144899) 96%	Early Access Review 16% Early Access Review 0% Other (5392421) 84%	-1 1	0 1
10	Counter-Strike	Ruined my life.	1	0
10	Counter-Strike	This will be more of a ''my experience with this game'' type of review, because saying things like '...	1	1
10	Counter-Strike	This game saved my virginity.	1	0

FIGURE 1: DATASET OVERVIEW

As illustrated in the figure, the dataset was highly unbalanced with a lot more positive reviews than negative ones. This makes sense the assumption is that Steam Games Library is the most popular distributor of games because they have a collection that people enjoy playing. The figure also clearly shows the problem with sarcasm in video game reviews. The first review says that the game ruined their life, yet they give the game a positive review. This can confuse most learning algorithms.

An auxiliary dataset containing a total of 78 columns with various tags about over 13000 different games from the Steam games store was also used [4]. However, I was only interested in the genre information of the games in the list.

B. DATA PREPARATION

As both datasets contained the game, its name as well as its unique steam app ID, I was able to join the information together to create a single dataset that had review text, sentiment score as well as genre

information. After looking at the distribution of word length for the data, I dropped all reviews that had a total length less than 50 words. I then cleaned and prepared two different sets of data for the different tasks. As there were very few missing values in the features I was interested in, my data was very complete.

I. Sentiment Analysis

As the data was very unbalanced, I used undersampling to create an even dataset that contained approximately 35,000 reviews. The dataset contained only the text features and the score of the review.

II. Multi-label Classification

I took the top games that had at least one true valued genre feature and created a dataset with approximately 75000 reviews and 50 different combinations of labels. I used random sampling to get 1500 reviews per genre configuration.

C. WORD EMBEDDINGS

Word Embeddings are a technique by which unlabeled word sequences can be mapped to a continuously valued low dimensional space that is able to capture the semantic and syntactic features of the original feature [5]. This mapping allows the text data to be used in machine learning algorithms in a meaningful and computationally efficient way. In my project I looked at a number of algorithms for generating this mapping. From the pre-trained variety, I use GloVe. I also use TF-IDF and Word2Vec which are trained using the dataset itself.

I. GloVe

The GloVe model efficiently leverages statistical information by training only on nonzero elements in a word-word co-occurrence matrix. It combines the advantages of two major model families already in the literature: global matrix factorization and local context window methods [6]. In my project I have implemented the model trained over 2 billion tweets and has 27 billion tokens.

II. Word2Vec

Word2Vec is a model architecture for computing vector representations of words from very large data sets. It is computationally very efficient compared to other techniques and can provide state of the art performance in measuring syntactic and semantic similarities between words [7]. In my project, I use the Continuous Bag of Words Model.

III. TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is an algorithm that used to calculate the weights of a set of words in some content and assign significance to each word in the set based on the frequency of appearance in the content offset by the frequency in the corpus.

For a word x in a document d, the weight $W_{x,d}$ of word x in document d is calculated by the following formula:

$$W_{x,d} = TF_{x,d} \log(\frac{N}{DF_x})$$

Where,
$TF_{x,d}$ – this is the frequency of occurrences of x in d
DF_x– this is the number of documents that contain x
N – this is the total number of documents

D. MODEL

I. LSTM

LSTM or Long Short-Term Memory networks are a type of recurrent neural network. They are capable of learning the significance of order in sequence prediction problems. In their research, Hochreiter et al. found it was able to solve complex time lag tasks that were previously unsolvable by any recurrent network algorithm [8]. In my project I use a modification of the LSTM proposed by Hochreiter and his colleagues called Bidirectional LSTM. The main idea behind a Bi-LSTM is to run inputs from two directions: one from the past to future and one from the future to the past. In Natural Language Processing tasks this technique is expected to allow the model to better understand context between words and semantics.

II. SVM

Support Vector Machines are training algorithms that maximize the margin between the training patters and the decision boundary [9]. The objective of the algorithm is to find a hyperplane in an N-dimensional space – where N is the number of features – that distinctly separates the data points.

E. EVALUATION METRICS

I. HAMMING LOSS

Hamming Loss is a metric often using in Multi-Label Classification problems. It is the fraction of labels that are incorrectly predicted.

If $\widehat{y_i}$ is the predicted value for the i-th label of a given sample, y_i is the corresponding true value, and n_{labels} is the number of classes or labels, then the Hamming loss $L_{Hamming}$ between two samples is defined as:

$$L_{Hamming}(y, \hat{y}) = \frac{1}{n_{labels}} \sum_{i=0}^{n_{labels}-1} 1(\widehat{y_i} \neq y_i)$$

where 1(x) is the indicator function.

II. PRECISION, RECALL, F-MEASURE

Precision is the ability of the classifier to not label as negative sample as positive and recall is the ability to successfully label all positive samples that are seen. F-measure (in this case F_1) is a weighted harmonic mean of precision and recall. In F_1, the recall and precision value are equally important. As I will also be performing multi-Label classification, I will also be using micro and averages for the labels. These are calculated as shown in Table 1.

TABLE 1: MULT-LABEL PRECISION, RECALL AND F-MEASURE METRICS [10]

average	Precision	Recall	F_beta
"micro"	$P(y,\hat{y})$	$R(y,\hat{y})$	$F_\beta(y,\hat{y})$
"samples"	$\frac{1}{\lvert S\rvert}\sum_{s\in S}P(y_s,\hat{y}_s)$	$\frac{1}{\lvert S\rvert}\sum_{s\in S}R(y_s,\hat{y}_s)$	$\frac{1}{\lvert S\rvert}\sum_{s\in S}F_\beta(y_s,\hat{y}_s)$
"macro"	$\frac{1}{\lvert L\rvert}\sum_{l\in L}P(y_l,\hat{y}_l)$	$\frac{1}{\lvert L\rvert}\sum_{l\in L}R(y_l,\hat{y}_l)$	$\frac{1}{\lvert L\rvert}\sum_{l\in L}F_\beta(y_l,\hat{y}_l)$
"weighted"	$\frac{1}{\sum_{l\in L}\lvert\hat{y}_l\rvert}\sum_{l\in L}\lvert\hat{y}_l\rvert P(y_l,\hat{y}_l)$	$\frac{1}{\sum_{l\in L}\lvert\hat{y}_l\rvert}\sum_{l\in L}\lvert\hat{y}_l\rvert R(y_l,\hat{y}_l)$	$\frac{1}{\sum_{l\in L}\lvert\hat{y}_l\rvert}\sum_{l\in L}\lvert\hat{y}_l\rvert F_\beta(y_l,\hat{y}_l)$
None	$\langle P(y_l,\hat{y}_l)\vert l\in L\rangle$	$\langle R(y_l,\hat{y}_l)\vert l\in L\rangle$	$\langle F_\beta(y_l,\hat{y}_l)\vert l\in L\rangle$

5. PERFORMANCE EVALUATION

I. SENTIMENT ANALYSIS

A. BASE MODEL

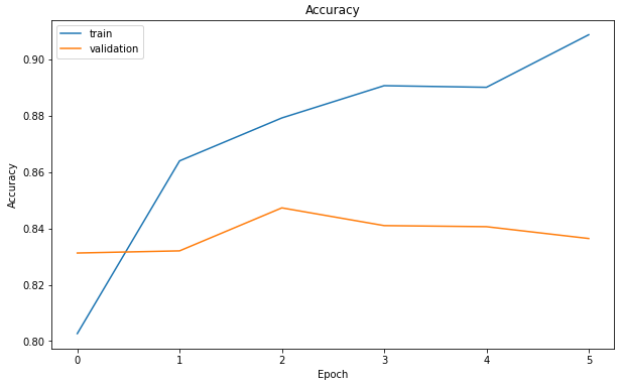

FIGURE 2: BASE MODEL ACCURACY

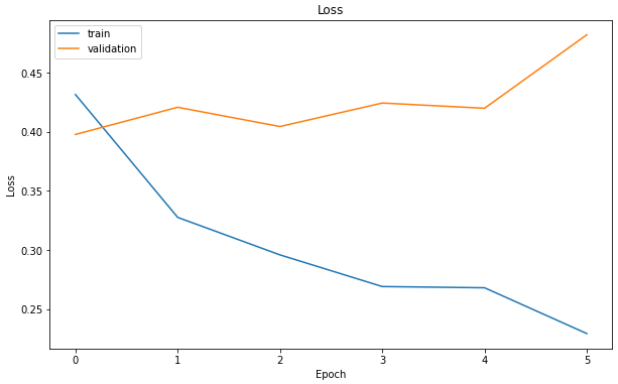

FIGURE 3: BASE MODEL LOSS

Test accuracy is 0.8416

	precision	recall	f1-score	support
0	0.82	0.88	0.85	11180
1	0.87	0.81	0.84	11320
accuracy			0.84	22500
macro avg	0.84	0.84	0.84	22500
weighted avg	0.84	0.84	0.84	22500

FIGURE 4: BASE MODEL METRICS

B. LSTM GLOVE

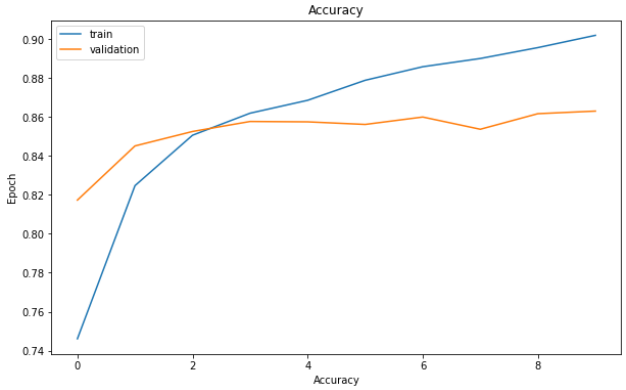

FIGURE 5: GLOVE MODEL ACCURACY

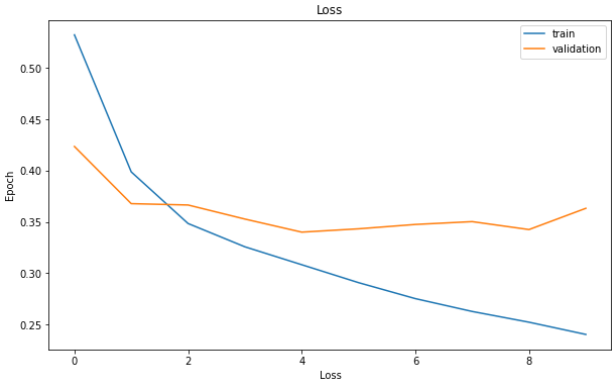

FIGURE 6: GLOVE MODEL LOSS

Test accuracy is 0.8643555555555555

	precision	recall	f1-score	support
0	0.87	0.85	0.86	11180
1	0.86	0.87	0.87	11320
accuracy			0.86	22500
macro avg	0.86	0.86	0.86	22500
weighted avg	0.86	0.86	0.86	22500

FIGURE 7: GLOVE MODEL METRICS

C. LSTM WORD2VEC

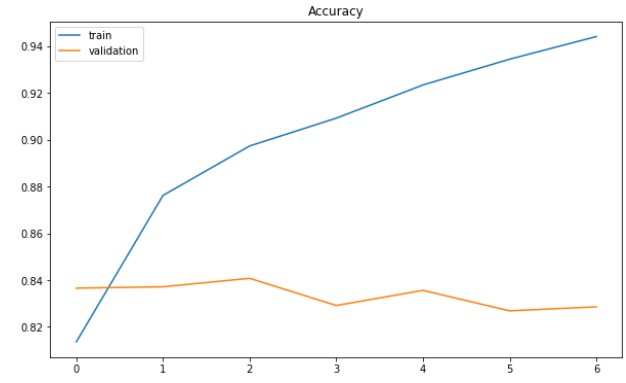

FIGURE 8: WORD2VEC MODEL ACCURACY

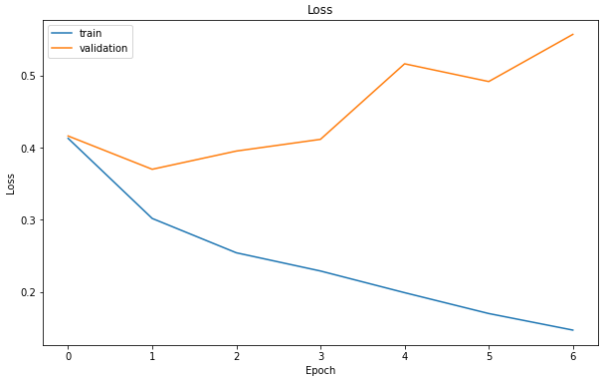

FIGURE 9: WORD2VEC MODEL LOSS

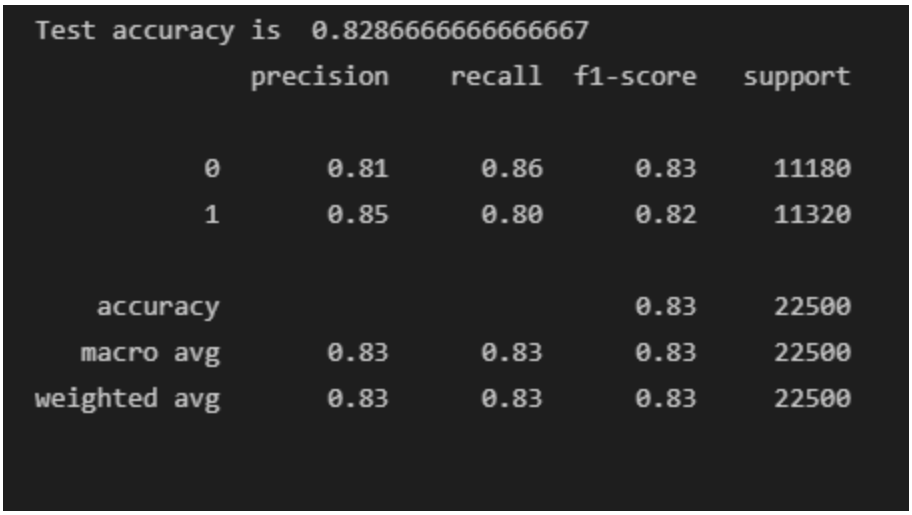

FIGURE 10: WORD2VEC MODEL METRICS

D. SVM TF-IDF

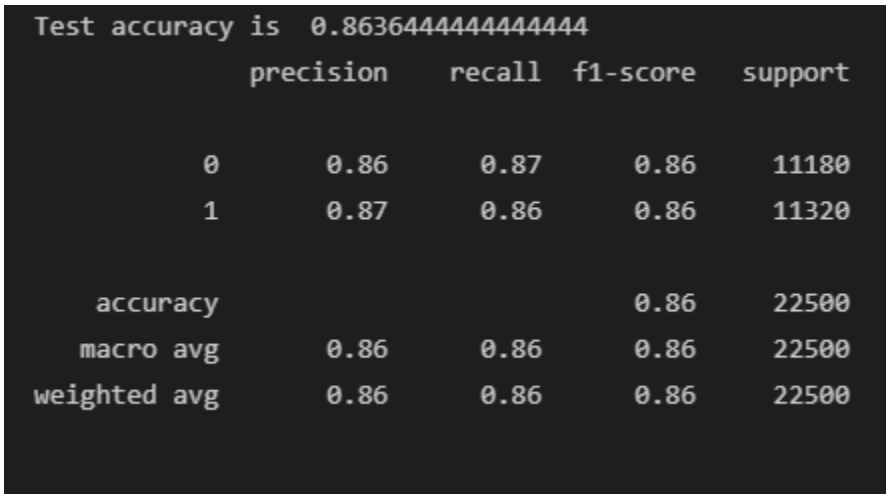

FIGURE 11: TF-IDF MODEL METRICS

II. MULTI-LABEL CLASSIFICATION

A. BASE MODEL

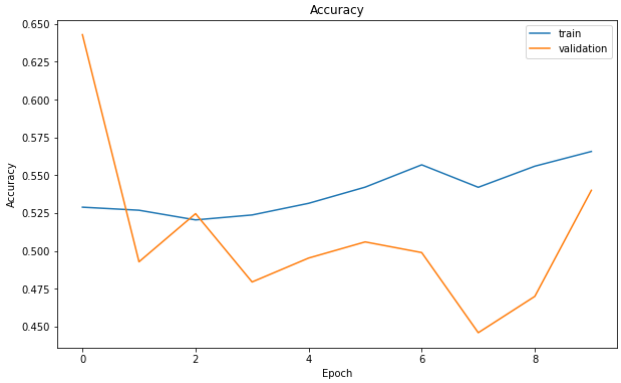

FIGURE 12: BASE MODEL ACCURACY

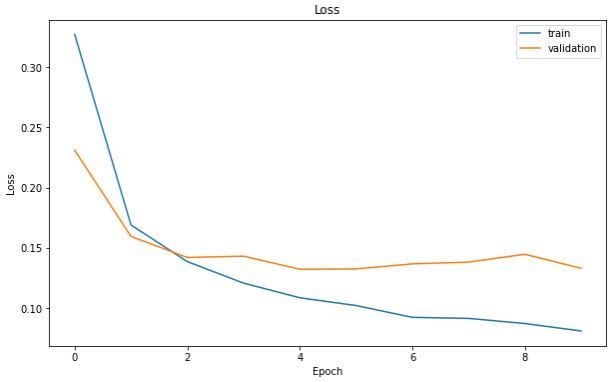

FIGURE 13: BASE MODEL LOSS

Hamming loss is 0.047104761904761906

	precision	recall	f1-score	support
0	0.93	0.94	0.94	12617
1	0.93	0.86	0.89	6802
2	0.89	0.84	0.87	4978
3	0.93	0.91	0.92	6725
4	0.92	0.89	0.90	4902
5	0.95	0.93	0.94	4511
6	0.95	0.92	0.93	4540
micro avg	0.93	0.90	0.92	45075
macro avg	0.93	0.90	0.91	45075
weighted avg	0.93	0.90	0.92	45075
samples avg	0.92	0.91	0.90	45075

FIGURE 14: BASE MODEL METRICS

B. LSTM GLOVE

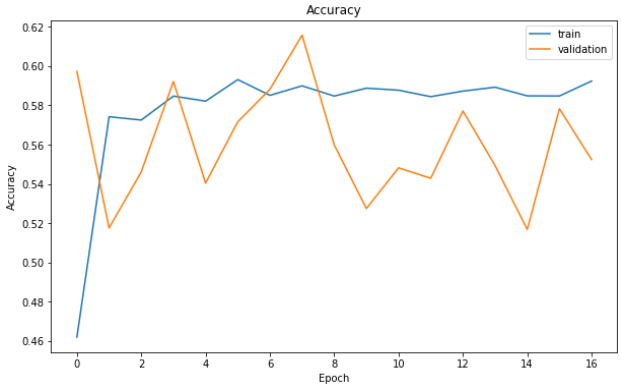

FIGURE 15: GLOVE MODEL ACCURACY

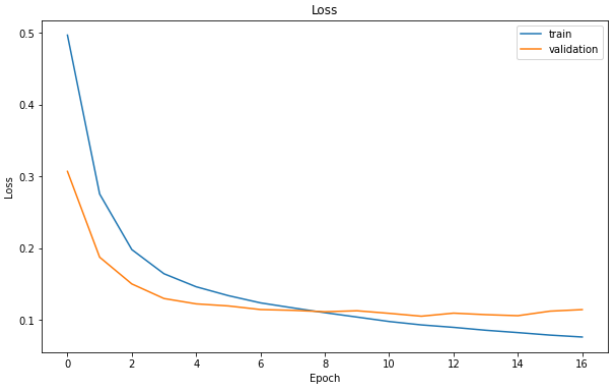

FIGURE 16: GLOVE MODEL LOSS

Hamming loss is 0.04245714285714286

	precision	recall	f1-score	support
0	0.95	0.94	0.94	12617
1	0.89	0.93	0.91	6802
2	0.92	0.85	0.88	4978
3	0.91	0.94	0.92	6725
4	0.92	0.90	0.91	4902
5	0.95	0.95	0.95	4511
6	0.96	0.93	0.94	4540
micro avg	0.93	0.92	0.93	45075
macro avg	0.93	0.92	0.92	45075
weighted avg	0.93	0.92	0.93	45075
samples avg	0.93	0.93	0.92	45075

FIGURE 17: GLOVE MODEL METRICS

C. LSTM WORD2VEC

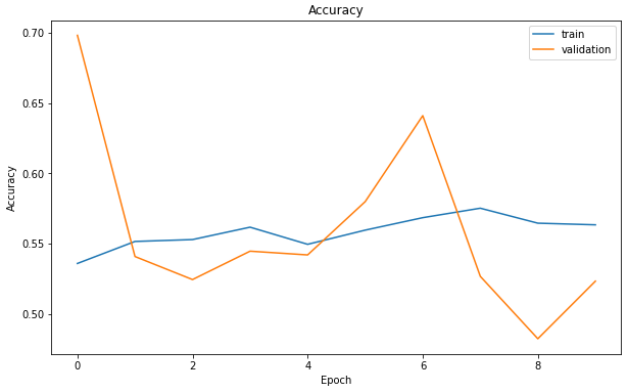

FIGURE 18: WORD2VEC MODEL ACCURACY

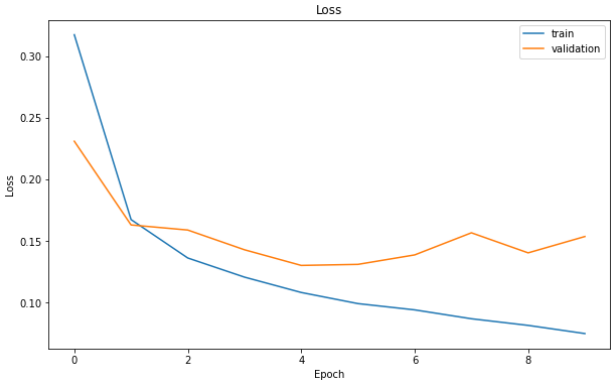

FIGURE 19: WORD2VEC MODEL LOSS

Hamming loss is 0.05370793650793651

	precision	recall	f1-score	support
0	0.95	0.90	0.93	12617
1	0.93	0.81	0.87	6802
2	0.90	0.81	0.85	4978
3	0.92	0.91	0.91	6725
4	0.88	0.90	0.89	4902
5	0.94	0.93	0.93	4511
6	0.96	0.88	0.92	4540
micro avg	0.93	0.88	0.90	45075
macro avg	0.92	0.88	0.90	45075
weighted avg	0.93	0.88	0.90	45075
samples avg	0.91	0.89	0.89	45075

FIGURE 20: WORD2VEC MODEL METRICS

D. SVM TF-IDF

Hamming loss is 0.04607619047619048

	precision	recall	f1-score	support
0	0.93	0.94	0.93	12617
1	0.92	0.89	0.90	6802
2	0.91	0.82	0.86	4978
3	0.94	0.90	0.92	6725
4	0.93	0.87	0.90	4902
5	0.97	0.92	0.94	4511
6	0.97	0.92	0.94	4540
micro avg	0.94	0.90	0.92	45075
macro avg	0.94	0.89	0.92	45075
weighted avg	0.94	0.90	0.92	45075
samples avg	0.92	0.90	0.90	45075

FIGURE 21: TF-IDF MODEL METRICS

6. CONCLUSION

From my results I found that the Stanford GloVe model pre-trained over tweets from Twitter.com performed better than other word embeddings techniques using Bidirectional LSTM. However, SVMs using TF-IDF were also very successful at both sentiment classification as well as multi-label genre classification. While the hamming loss is extremely low, and the f1-scores are much higher than other work it is important to note that the model is only training and validating over 50 out of a possible 2^7 possible label combinations.

An interesting result I found was that in my implementation I had run my multi-label models using the same data as the set I used in sentiment analysis. Here, there was random sampling and so any combination of the 2^7 were present. This model performed worse as expected however it notable predicted a specific review it got correct the first time incorrectly. In future work I would like to explore how to increase the knowledge of the model (the number of labels it can classify) without confusing it's previous knowledge.

REFERENCES

[1] L. Z. Yuhang Jiang, “Deep learning for video game genre classification,” 21 November 2020. [Online]. Available: https://arxiv.org/pdf/2011.12143.pdf.

[2] D. L. A. H. C.-P. B. Markos Viggiato, “What Causes Wrong Sentiment Classifications of Game Reviews?,” *IEEE Transactions on Games,* 2021.

[3] Larxel, “Steam Reviews | Kaggle,” 2021. [Online]. Available: https://www.kaggle.com/andrewmvd/steam-reviews. [Accessed 3 11 2021].

[4] C. Kelly, “Steam Game Data - dataset by Craig Kelly | data.world,” 2016. [Online]. Available: https://data.world/craigkelly/steam-game-data. [Accessed 3 11 2021].

[5] T. Y. Yang Li, Guide to Big Data Applications, Springer International Publishing, 2017.

[6] R. S. C. D. M. Jeffrey Pennington, “GloVe: Global Vectors for Word Representation,” 2014.

[7] K. C. G. C. J. D. Tomas Mikolov, “Efficient Estimation of Word Representations in Vector Space,” 2013.

[8] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation,* pp. 1735-1780, 1997.

[9] I. M. G. V. N. V. Bernhard E. Boser, “A Training Algorithm for Optimal Margin Classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*, Pittsburgh, 1992.

[10] “3.3. Metrics and scoring: quantifying the quality of predictions - scikit-learn.org,” 2013. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#hamming-loss. [Accessed 12 12 2021].