732A92 - Text Mining, Project Report

Predicting Videogame Recommendations

Comparing LSTM and DistilBERT on Steam reviews

Otto Moen (ottmo242)

Abstract

The video game industry has been growing rapidly with more and more games being produced every year. In an ever more crowded market, it can therefore be difficult for consumers to know which games are worth their time and money. On the distributing platform Steam, users can leave reviews and tag them as "recommend" or "not recommend" but this is not the case on other platforms. If such reviews could be automatically tagged it could potentially allow for aggregation of reviews from many sites to aid people in their choices. To achieve this a bidirectional LSTM as well as a DistilBERT model was trained on a data set of Steam video game reviews to predict if a review recommends a game or not. Compared to a Naive Bayes model with an accuracy of 0.82 as a baseline, both the LSTM and DistilBERT models achieved an accuracy of 0.83, which was lower than expected. One likely reason for this was a too simplistic preprocessing of the raw texts. A limited amount of computing resources also slowed the process of training DistilBERT models, such that only a few designs could be tried out.

Contents

List of Figures

List of Tables

1. Introduction

Over the past few decades, video games have grown to become an important part of the culture in many countries. While in the past gaming was mainly seen as an activity for children along with their other pastimes, it is now something that people of all ages take part in, often for large amounts of time every week. Gaming is no longer simply a diversion, with the rise of eSports and content creation on sites such as Youtube and Twitch opening up many opportunities for serious careers centered around it. The past two years have also seen the impact of COVID-19, with many people forced to work from home and losing access to their regular pastime activities. As such the video game industry has been growing rapidly with more and more games being produced every year (Clement, 2021). In an ever more crowded market, it can therefore be difficult for consumers to know which games are worth their time and money.

Some platforms for distributing games, such as the largest one, Steam by Valve, allow users to leave reviews about the games they have played. On Steam users also tag their review with "recommend" or "don't recommend" which makes it easier to get an overall picture, but this is not always the case on the many platforms and forums in which games are discussed. For instance, the platform by Epic Games, which has grown from 32 million monthly active users in 2019 to 56 million in December 2020, does not offer user reviews (Epic Games, 2021). With more people playing games on other platforms than Steam it follows that the reviews for these games will also be more spread out, making it harder for consumers to find a good source of information when buying games. However, if it were possible to aggregate reviews from many sources and correctly classify them automatically, this would lower the impact on players that the increased competition might otherwise have.

To solve this problem a Long short-term memory (LSTM) model was trained on a set of Steam game reviews to classify whether the review was recommending or not recommending others to buy a game. The model achieved an accuracy of 0.83 on the test set which is a very slight improvement upon more basic models such as Naive Bayes which had an accuracy of 0.82. Another state-of-the-art model, Distilled Bidirectional Encoder Representations from Transformers (DistilBERT), was likewise trained and it also reached an accuracy of 0.83.

2. Theory

In this chapter the relevant theory used for estimating and evaluating the used models will be presented; including the Naive Bayes model, the LSTM model as well as BERT.

2.1 Naive Bayes

The Naive Bayes classifier is a simple and widely used probabilistic model which is able to capture the uncertainty of the predictions with posterior distributions (Tan et. al., 2019). The classifier assumes that all attributes $\mathbf{x}$ are conditionally independent of one another given the class label, which greatly simplifies the probability calculations. With the assumption, the posterior probability for a test instance can then be computed as

$$P(y|\boldsymbol{x}) = \frac{P(y)\prod_{i=1}^{d} P(x_i|y)}{P(\boldsymbol{x})} \tag{2.1}$$

where d is the number of attributes. The denominator acts as a normalizing constant and can therefore be dropped from the equation, meaning it is sufficient to choose the class that maximizes the numerator. The simplicity of the Naive Bayes classifier can however be a drawback as in many scenarios the attributes are not conditionally independent, which will negatively impact the performance of the model. As such it is often used as a baseline against which more complex models are compared.

For text data, a Naive Bayes model requires the raw texts to be processed into a matrix where each column corresponds to a word/term and each row corresponds to a document. The values in the matrix can then for example be the number of times a word occurs in a document or a numerical statistic such as the term frequency-inverse document frequency (TF-IDF).

2.2 LSTM

A recurrent neural network (RNN) is a type of neural network that is designed to handle data with temporal dependence, such as text or speech data (Abiodun et al., 2018). This is done by introducing cycles, or loops, meaning the information does not only flow forward through the network but is persisting in the nodes. This has been referred to as short-term memory. One issue with RNNs however is that knowing *what* to store in the short-term memory is a very complicated challenge, one which traditional algorithms do not solve very well or take too much time solving (Hochreiter & Schmidhuber, 1997).

To solve this challenge Hochreiter & Schmidhuber (1997) introduced the gradient-based method "Long Short-Term Memory" (LSTM). The LSTM model introduces memory cells which are special, self-connected, units within the network designed to store information across time. Adding on to this, a memory cell is enclosed by two other units; the input gate and output gate. These two units control what information flows in and out of the memory cell, protecting it from irrelevant information but also protecting the rest of the network from the information in the memory cell if it is not currently relevant. As with the rest of the network, the two gates have parameters that are learned when the network is being trained such that they will learn when to open and close.

The memory cells are often grouped together into so-called memory cell blocks. Instead of each memory cell having its own input and output gate a block of cells share the same gates. This is done to increase performance, allowing for storage of more complicated information that could be coded in a single cell, as well as efficiency, by reducing the number of units. Hochreiter & Schmidhuber (1997) provide an example of how such a net could look, which is presented in figure 2.1.

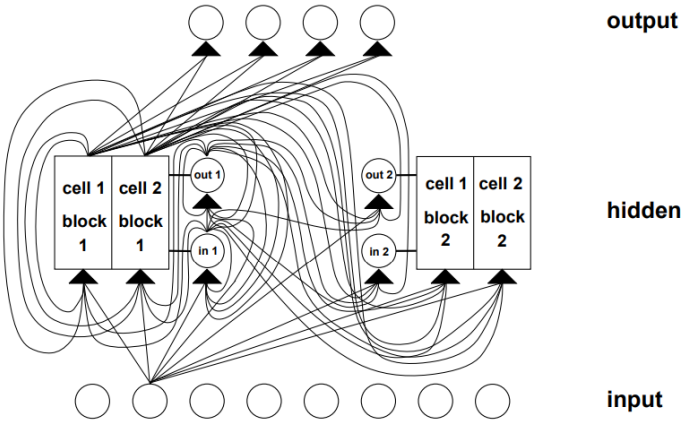

Figure 2.1: Example of an LSTM model with 8 input units, 4 output units and 2 memory cell blocks of size 2.

An extension to the LSTM model is the Bidirectional LSTM. This is essentially just two LSTM models combined, where the second model trains on the reversed input sequence, i.e. back-to-front. The two models are then merged with one of the following methods:

- Sum
- Multiplication
- Averaging
- Concatenation

with the default method being concatenation.

2.3 GloVE

An alternative method to preprocessing the data to term frequency or TF-IDF is word embedding. This method consists of converting each word to a vector of numeric values with the idea that this representation should capture semantic regularities (Pennington et al., 2014). This embedding process can be implemented as an initial layer in a neural network, where the vector values will be treated as parameters to be optimized along with the rest of the network. This process can however be quite time-consuming depending on the size of the vocabulary one is working with. Another issue can be that if a word in the vocabulary occurs very few times in the texts used for training the model will not have many opportunities to update the vector values.

An alternative to training the word vectors from scratch is to instead import them from a pre-trained model. These vectors can then be used either as-is or as initial values which are to be fine-tuned on the specific data at hand. One such collection of pre-trained vectors is the Global Vectors for Word Representation (GloVE) introduced by Pennington et. al. (2014). Since the first release, they have produced several different variants of GloVE based on different sources of text such as Wikipedia or Twitter. For each source, there are also vectors of different lengths, ranging from 25 up to 300.

2.4 BERT

Rather than only providing pre-trained word vectors used for embedding there are also models with more extensive architecture pre-trained on general texts. One such model is the language representation model Bidirectional Encoder Representations from Transformers, or BERT (Devlin et al., 2019). It is a deep network with many millions of parameters. These parameters can either be used directly by attaching them to an output layer or they can be fine-tuned on data. Additionally, it is also possible to add more layers in between BERT and the output layer to further tune the model to a specific task which the general pre-training might not perform as well at.

2.4.1 DistilBERT

As was mentioned the BERT model is a very deep network with many parameters (110 million for BERT-base and 340 million for BERT-large. As such it can become a difficult challenge to use these models, in terms of computing resources, particularly if the data set they are to be fine-tuned on is also large. In an attempt to solve this issue Sanh et al. (2019) used knowledge distillation, meaning when a smaller model attempts to replicate the output of a larger model, to create a model they named DistilBERT. The Distilbert model contains 66 million parameters, which represents a 40 percent reduction in size. The authors showed that this model was able to run 60 percent faster while retaining 97 percent of the language understanding capabilities.

3. Data

The data used in this report is a set of 17,494 Steam game reviews obtained from the Machine Learning website Kaggle.com (Möbius, 2021). Each data point consists of the game title, the year the review was posted, the raw review text, and if the reviewer recommends or does not recommend others buy the game. The recommendation is coded to 1 if the user recommends it and 0 if the user does not recommend it. This recommendation variable is used as the class variable and is what the models will be trained to predict. The distribution of the two classes is shown in figure 3.1.

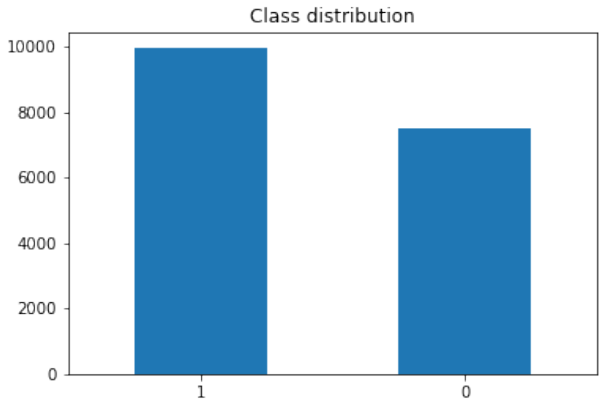

Figure 3.1: Number of reviews recommending or not recommending a game

As shown the data set contains more observations of class 1 (9,968) compared to class 0 (7,526) but with proportions of roughly 57 vs 43 percent it is not deemed to be very unbalanced. The raw text was preprocessed by tokenizing it; removing stop words and only keeping words consisting of alphabetic letters. An example from the data set would be the sentence *"Not many games have the cute tag right next to the horror tag on Steam"* which after the preprocessing became the following string: *"games cute tag right horror tag steam"* To investigate further the lengths of these lists are shown in a histogram in figure 3.2.

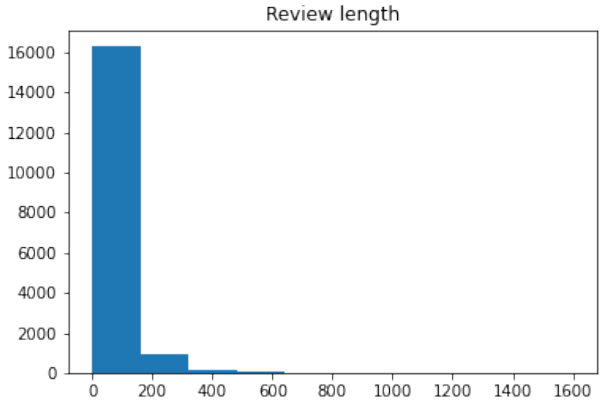

Figure 3.2: Number of words per review when splitting on space

It can be observed that a large majority of the reviews contain between 1 and 200 words but that there are reviews up to 1600 in length. However, only 3 reviews in total are longer than 1000 words and only 5 are longer than 700. As the reviews will all be padded up to the maximum length when preparing the data for the LSTM model, including these 5 observations would make the data needlessly large and negatively impact training times. For this reason, they are removed. Lastly to get an overview of the contents of the reviews the 20 most common words are extracted and shown in figure 3.3.

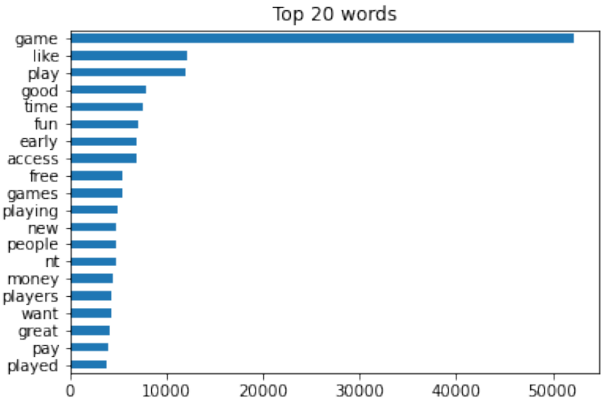

Figure 3.3: Number of words per review when splitting on space

The word **game** stands out as the by far most common word, occurring over 50,000 times compared with the second most common word **like** with only around 15,000 occurrences. However, do err on the side of caution it was not deemed a domain-specific stop word and was thus kept. With the preprocessing done the data set was divided into three sets; training, validation, and test consisting of 80, 10, and 10 percent respectively.

4. Method

4.1 Naive Bayes

Initially, the Naive Bayes model was estimated in order to have a baseline from with to compare performances of the LSTSM model. The preprocessed text of the training set was vectorized with a TF-IDF vectorizer and this matrix of TF-IDF values was then used to fit a Naive Bayes Model. As was observed in figure 3.1 in chapter 3 the classes were not very unbalanced, but even so, a balanced version of the training set was also created using random oversampling for comparison. Lastly, different parameter values were tested with a grid search, using cross-validation with five folds, to find the optimal settings for this data. The parameters that were tested for were if the TF-IDF vectorizer should have *binary* set to *true* or $false$ and if it should have $ngram\_range$ set to $(1, 1)$ or $(1, 2)$. For the Naive Bayes classifier, the parameter that was tested for was the smoothing parameter α, with possible values ranging from 0.1 to 1 in steps of 0.1.

4.2 LSTM

For the LSTM model, the texts were first encoded, turning the strings into sequences of numbers. These sequences were then padded to the length of the longest sequence. An embedding matrix was then created using the pre-trained GloVe vectors. The Twitter collection of GloVe vectors was chosen with vector dimension 200. The Twitter vectors were chosen ahead of the Wikipedia vectors due to the data. Game reviews mainly consist of the opinions of the reviewer. It would therefore seem likely that such texts are more similar to tweets, where people often write their opinions about something, rather than Wikipedia which should be more based on facts. The vector dimension of 200 was chosen as it was the largest one available.

With the embedding matrix in place, everything was set up to begin designing the network. Much experimentation was carried out, on both the initial training set as well as the balanced version, to find the model which performed the best on the validation set. The final model was surprisingly simple. From the input layer, the observations were fed into an embedding layer with the weights set as the embedding matrix and the option to train these further turned off (trainable = false). From the embedding layer, the data was then fed into a bidirectional LSTM where each directional LSTM had 64 units for a total dimension of 128. To prevent overfitting a dropout layer was added with a dropout rate of 0.6. This was followed by a fully-connected layer with 64 hidden units, using ReLU as an activation function. From this, the data went through another dropout layer with a dropout rate of 0.5, before finally entering the output layer, which used a sigmoid activation function. The model is visualized in figure 4.1.

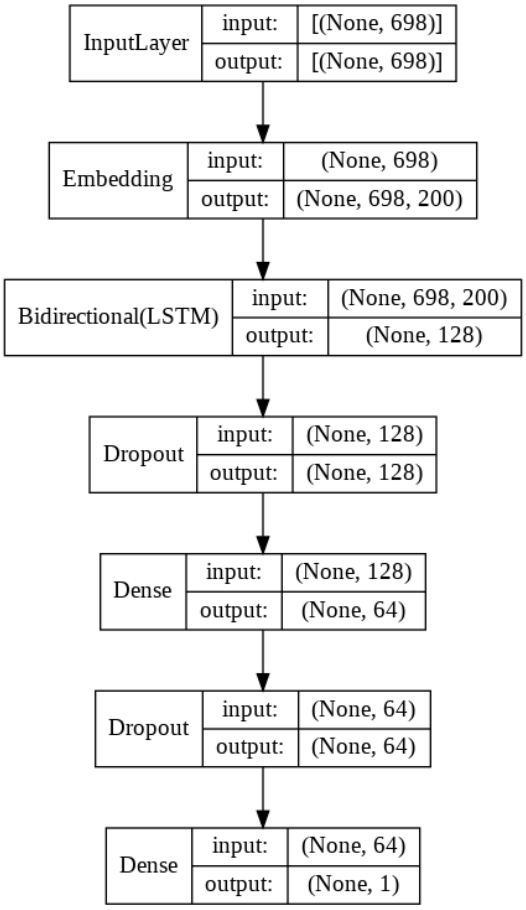

Figure 4.1: Overview of the final LSTM model

This results in a model with a total of 7,886,801 parameters. However, 7,742,800 of these belong to the embedding layer and are therefore not trainable, which means that the number of parameters that have to be trained amounts to just 144,001. The model was then compiled with the Adam optimizer and using binary cross-entropy as the loss. The compiled model was then fitted on the training set using a batch size of 256.

4.3 DistilBERT

At first, an attempt was made to train a model using BERT but faced many issues, the main one being that the Google Colab GPU server was unable to train the model due to a lack of RAM. To solve this issue a switch was made from BERT to the smaller DisilBERT model. The procedure for the DistilBERT model was similar to that of the LSTM. First, the texts of the three datasets were encoded using the pre-trained DistilBERT encoder. One difference compared to the LSTM model is that the BERT models have two inputs; the encoded texts as well as an attention mask, which are vectors paired with the encoded texts where the values are 1 if the value in the same position represents an actual word and 0 if the value is there due to padding. However, even with DistilBERT being smaller than BERT, using the maximum review length of 698 still led to memory issues when trying to train the model. As was seen in chapter 3 the vast majority of reviews were between 0 and 200 tokens in length and an even greater proportion if going up to 300 words. The maximum length was therefore lowered to 300, which reduced the size of the data enough so that it could be trained.

The model was then designed to find the one that performed the best on the validation set. For the final model, the inputs were first fed into the pre-trained DistilBERT model where the parameters were allowed to be further fine-tuned, i.e. not frozen. This was followed by a dropout layer with a dropout rate of 0.15, after which followed a fully connected layer with 768 hidden units and a ReLU activation function. This pattern was repeated a second time, meaning a second dropout layer and a second fully connected layer. A third dropout layer was added and then finally an output layer with a sigmoid activation function as well as a small amount of L2 regularization. The model is visualized in figure 4.2.

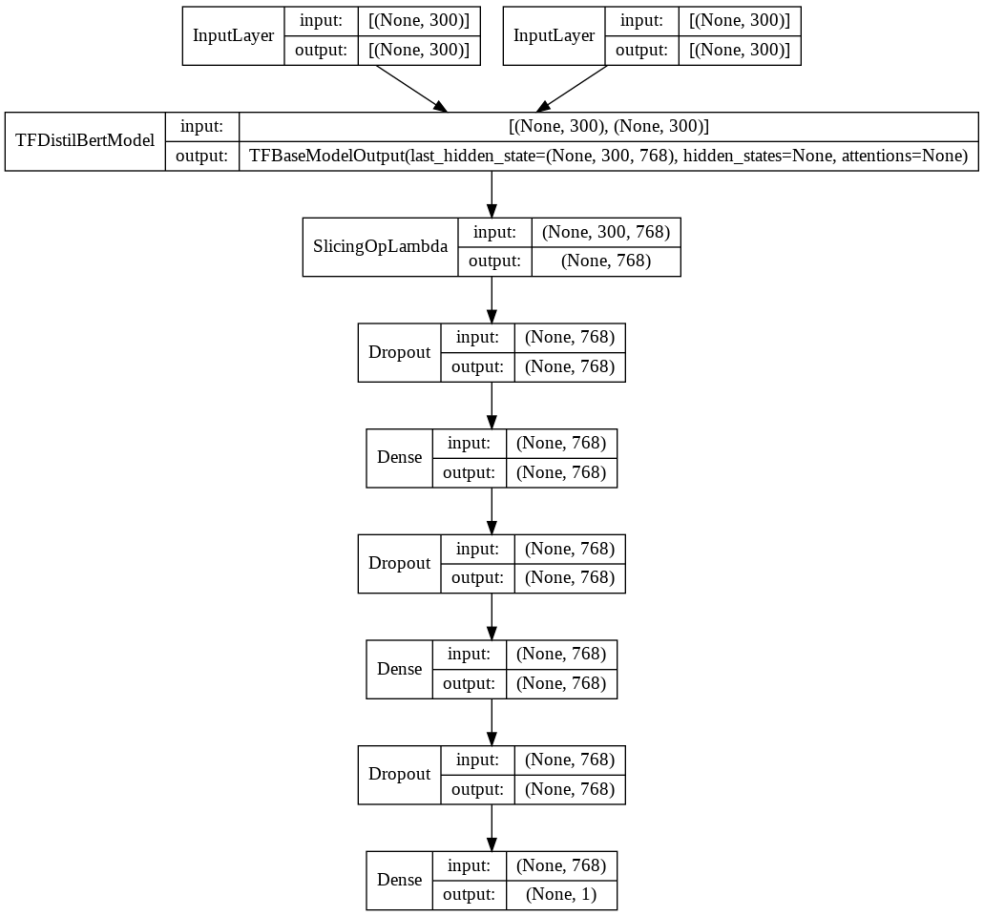

Figure 4.2: Overview of the final DistilBERT model

This results in a model with a total of 67,544,833 parameters. Since the DistilBERT model was set to be fine-tuned, unlike for the LSTM model, all the parameters were trainable. The model was then compiled with the Adam optimizer and using binary cross-entropy as the loss. The learning rate for the Adam optimizer was set to $5e^{-5}$ as recommended by the authors of DistilBERT so as to not overwrite too much of the pre-trained model weights. The compiled model was then fitted on the training set using a batch size of 32 to not run out of memory.

5. Results

5.1 Naive Bayes

As a starting point, a Naive Bayes model was fitted on the training set with the default values and then used to predict on the validation set, the results of which are presented in table 5.1 and figure 5.1.

Table 5.1: Results of Naive Bayes, default settings

	precision	recall	f1-score	support
0	0.89	0.59	0.71	1093
1	0.77	0.95	0.85	1525
accuracy			0.80	2618
macro avg	0.83	0.77	0.78	2618
weighted avg	0.82	0.80	0.79	2618

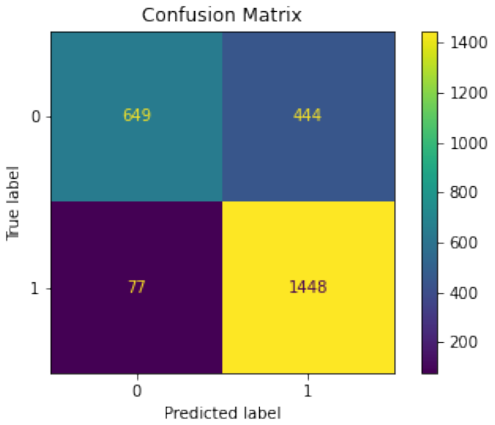

Figure 5.1: Confusion matrix with default settings

From the results, it can be observed that the overall accuracy of the model on the validation data is 0.80. The recall for positive reviews is high (0.95), with only 77 reviews incorrectly classified as negative, but the precision is lower (0.77) and for the negative reviews the situation is the opposite. In particular, the recall for negative reviews is much lower at 0.59. As was mentioned in chapter 4 a balanced data set was then created with random oversampling and used to fit a Naive Bayes model. The results of the predictions on the validation set are presented in table 5.2 and figure 5.2.

Table 5.2: Results of Naive Bayes, default settings & balanced training set

	precision	recall	f1-score	support
0	0.75	0.84	0.79	1093
1	0.87	0.80	0.84	1525
accuracy			0.82	2618
macro avg	0.81	0.82	0.81	2618
weighted avg	0.82	0.82	0.82	2618

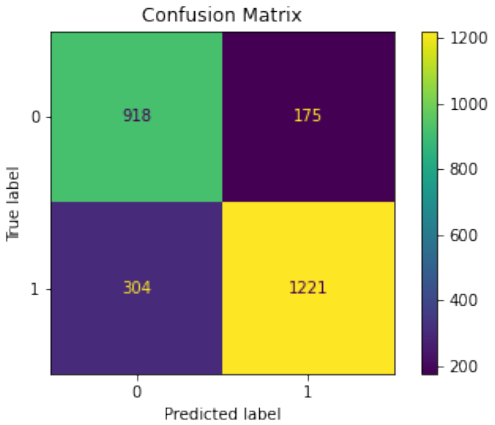

Figure 5.2: Confusion matrix with default settings & balanced training set

As can be observed balancing the training set leads to a small improvement in accuracy, up to 0.82. The gaps between precision and recall that existed previously are also smaller. The f1-score for the positive reviews decreased somewhat to 0.83, but for the negative reviews, the score improved from 0.71 to 0.79. As the performance on the negative reviews benefited from the balancing of the training set while the positive reviews only suffered slightly the balanced training set was the one optimized further.

The grid search over the different parameter values resulted in the TF-IDF vectorizer having $binary = True$, which for this vectorizer does not mean that the whole output is binary but only that the TF term is binary. The vectorizer also ended up having $ngram\_range = (1, 1)$ meaning only unigrams were used. For the classifier, the optimal smoothing parameter was $\alpha = 0.2$. This model was then used to predict on the validation set, resulting in the same accuracy of 0.82. This was chosen as the final model and then used to predict on the test set, the results of which are presented in table 5.3 and figure 5.3.

Table 5.3: Results of Naive Bayes on test set, optimal settings

	precision	recall	f1-score	support
0	0.84	0.71	0.77	1133
1	0.80	0.90	0.85	1489
accuracy			0.82	2622
macro avg	0.82	0.80	0.81	2622
weighted avg	0.82	0.82	0.81	2622

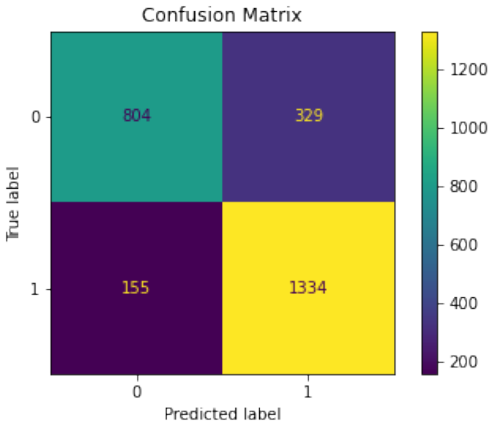

Figure 5.3: Confusion matrix on the test set, Naive Bayes with optimal settings

The accuracy on the test set is 0.82, which is the same as for the validation data. The recall for the positive reviews and the precision for the negative reviews is high as can be observed in the confusion matrix where only 155 positive reviews were incorrectly classified as negative. On the other hand, 329 negative reviews were classified as positive.

5.2 LSTM

As LSTM model with the design described in chapter 4.2 was fitted on the balanced training set. The result of the training process is shown in figure 5.4.

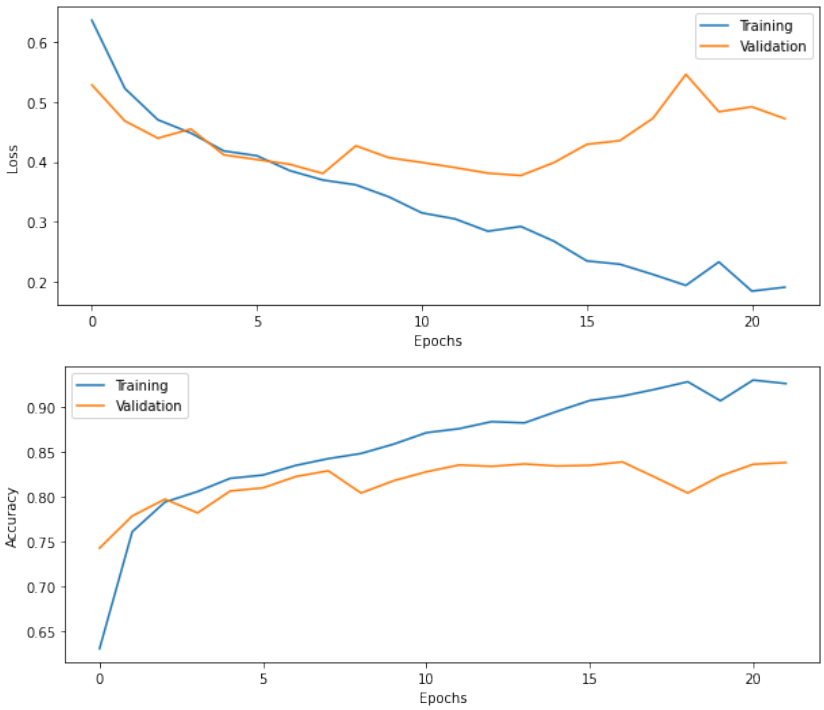

Figure 5.4: Loss and accuracy for each epoch

As can be observed the validation loss decreases slightly at first and is then stable for a while before beginning to increase. Similarly, the validation accuracy follows the training accuracy quite well for about the first 7 epochs, after which it does not improve much. The epoch during which the model had the highest validation accuracy (0.84) was saved, and these settings are then used to predict on the test set, the results of which are presented in figure 5.4 and figure 5.5.

Table 5.4: Results of LSTM on test set

	precision	recall	f1-score	support
0	0.80	0.80	0.80	1133
1	0.85	0.84	0.85	1489
accuracy			0.83	2622
macro avg	0.82	0.82	0.82	2622
weighted avg	0.83	0.83	0.83	2622

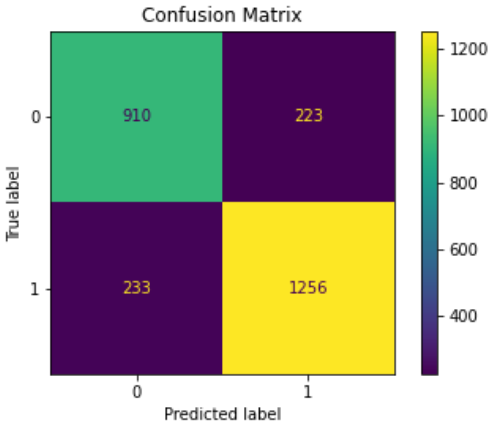

Figure 5.5: Confusion matrix on the test set, LSTM

The overall accuracy of the LSTM model on the test set is 0.83. The precision for the positive reviews is the highest at 0.84, with the other values slightly lower. As can be observed in the confusion matrix there were 223 wrongly classified negative reviews and 233 wrongly classified positive reviews.

5.3 DistilBERT

A DilstilBERT model with the design described in chapter 4.3 was fitted on the training set. The result of the training process is shown in figure 5.4.

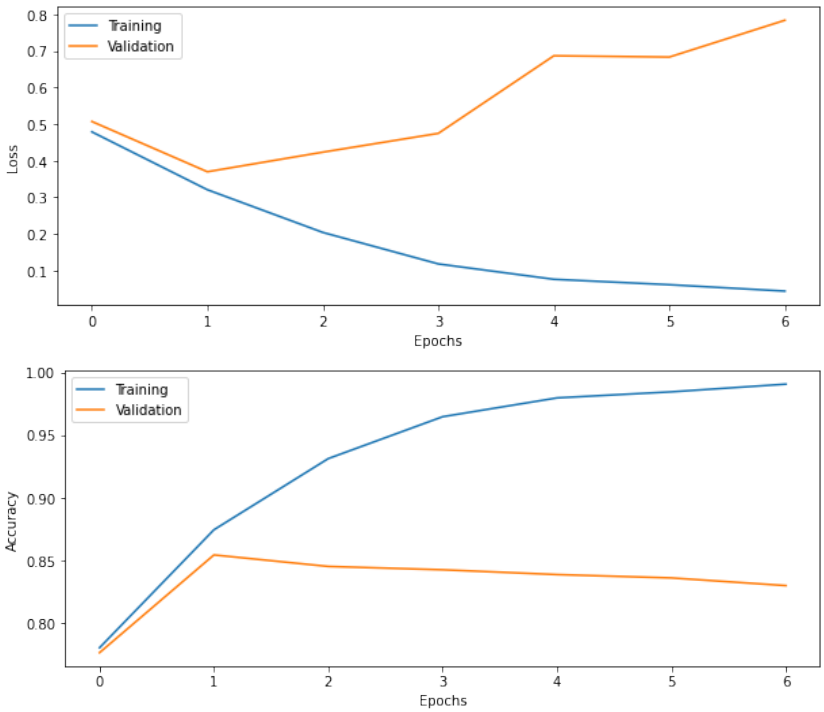

Figure 5.6: Loss and accuracy for each epoch

As can be observed the validation loss decreases from the first to the second epoch, after which it starts increasing. Similarly, the validation accuracy only improves from the first to the second epoch, after which it starts declining. The epoch during which the model had the highest validation accuracy (0.85) was saved, and these settings are then used to predict on the test set, the results of which are presented in figure 5.5 and figure 5.7.

Table 5.5: Results of DistilBERT on test set

	precision	recall	f1-score	support
0	0.85	0.73	0.79	1133
1	0.82	0.90	0.86	1489
accuracy			0.83	2622
macro avg	0.83	0.82	0.82	2622
weighted avg	0.83	0.83	0.83	2622

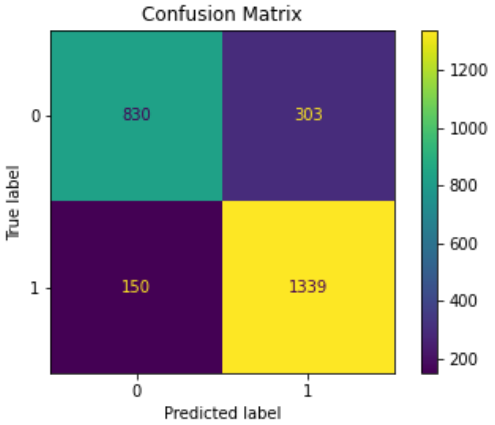

Figure 5.7: Confusion matrix on the test set, DilstilBERT

The overall accuracy of the DistilBERT model on the test set is 0.83. The recall for the positive reviews is the highest at 0.90, with the recall for the negative reviews the lowest at 0.73. As can be observed in the confusion matrix there were 303 wrongly classified negative reviews versus only 150 wrongly classified positive reviews.

6. Discussion

As was seen in the result the LSTM and DistilBERT (0.83 on the test set) models did not have much in the way of better performance compared to the Naive Bayes model, which is surprising. One likely reason is simply that the limited time and scope of this project meant that potential models which would have performed better were just not found. It was particularly difficult to find a good DistilBERT model. Even with the fewer parameters compared to BERT, and using a GPU in Google Colab, training a model could often take a long time, which limited the number of models that could be trained. Google Colab also has user limitations on how much their GPU servers can be used, which meant that some days GPU was not available for use at all.

However, that reason is beyond this discussion, other than the straightforward conclusion that more time and computing power would have aided in finding better models, in particular a better DistilBERT model. The discussion will instead focus on more feasible aspects that can have negatively impacted the performance. This was done by investigating the observations that were incorrectly classified by both the LSTM and DistilBERT models. This gave rise to two potential issues.

The first one is that some of the reviews are quite complicated in terms of having both positive and negative words. Some reviews are mostly positive, but with one or two lines where the user explains why despite everything just said they can still not recommend the game, or vice versa. Two examples of incorrect reviews, one of each class, are shown below.

Positive review with negative prediction:
I love the concept behind this game and the way it works and looks. What I'm not too thrilled about is how SOE no longs makes any of the items in the store and lets the players do all the work for them. The Vanu Faction is easily more powerful than the other two factions while the Terran Republic is the weakest faction based on stats alone. The prices are too high and every update/hotfix tends to break more things than they fix. My final line: Download the game and try it out, but don't put money into it until SOE gets it together and stops making all these mistakes.

Negative review with positive prediction:
You know I played this game for like two days wiht my friend and couldnt stop. Until the next day when I tried to get on again the game kept... crashing? I dont acutally know whats happening. But from what ive played and what i've been reading this game USED to be good.

As can be observed the positive review does have a lot of negative aspects to it, criticizing many areas of the game. Similarly the negative review has some positive elements and mainly the one negative line about crashing. These sorts of texts likely had a negative impact on model performance.

The second potential reason, which after investigation seems to have had quite a big effect on prediction errors, is the preprocessing of the raw texts. It was carried out with a fairly simple function which upon review does not seem enough to handle the complexity of the data. When comparing the raw texts of the misclassified reviews and their processed counterpart some of these examples stood out clearly. One such is shown below and discussed.

Raw review:
Don't quit just when you start because you dislike the game at first I made that mistake. Just slowly learn the mechanics and you'll be fine. Also don't let your friends make you waste your money on the Mann co store(I also made that mistake). It is a great game.

Processed review:
quit start dislike game mistake slowly learn mechanics fine let friends waste money mann co mistake great game

This review is one that was recommending players to buy a game but was classified as not recommending. It can be observed that the raw text is overall a positive one, but one which contains many negative words. Unlike in the previous example, the issue here is that when the stop words were removed what was left was the largely negative words, such as "dislike game" and "waste money," which would explain why it was wrongly classified. One more example of a negative review that was misclassified as positive, where the preprocessing has gone rather hilariously wrong, is shown below.

Raw review:
ARGHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHHHHHHHHHHHHHHHHHHHHHHHHHH HHH!!! !! !! !!!!!!IT ALWAYS CRASHES!!!!!Other than that its kinda fun... still... :/

Processed review:
kinda fun

Clearly a more comprehensive preprocessing would have been prudent to avoid mistakes such as these. When comparing with other work on classifying video game reviews, it does indeed seem that how the raw text is processed can have a large effect on the model performance. For instance, in terms of the Naive Bayes classifier using TF-IDF values as input, which in this report achieved a test accuracy of 0.82, a wide range of accuracies were found. Examples include Bais et al. (2017) who reached an accuracy of 0.71, Roy et al. (2018) with 0.89, and Muller (2021) with 0.87. It should be noted that unlike Bais et al. (2017) and Muller (2021), Roy et al. (2018) were not using reviews from Steam and as such the general style of their reviews might differ some. All three works used TF-IDF values as the inputs. An example where only keywords were used, Reddy et al. (2016) achieved an accuracy of just 0.64.

For the LSTM model, there were not as many works easily available for comparison. Muller (2021) also implemented an LSTM model for Steam reviews with an accuracy of 0.89, two percent better than their Naive Bayes model. Compared to this report, where the LSTM outperformed the Naive Bayes by one percent, it does seem that it might be a difficult challenge to design an LSTM that would greatly outperform the simpler Naive Bayes model.

7. Conclusion

In this project, a dataset consisting of Steam video game reviews was used to train three models in an attempt to correctly predict whether a review was recommending other users buy the game or not. At first, a more simple Naive Bayes model was trained to use as a baseline and achieved a test accuracy of 0.82. Secondly, a bidirectional LSTM using GloVEs pre-trained word embedding vectors was trained, reaching a test accuracy of 0.83. Finally, a model using the pre-trained DistilBERT as initial weights, which were then fine-tuned on the data, was trained, also achieving a test accuracy of 0.83.

Based on the discussion of the results, it is likely that the performance of all models was negatively impacted by the simple methods used to preprocess the data. This was also observed when comparing with similar works on game reviews, where the test accuracies were quite varied despite similar raw data and similar models. For future projects, it is clear that more attention should be placed on processing the raw text. This is especially the case when the data set consists of such a varied collection of texts. Compared to a cleaner and more homogeneous set, for example news articles, the data set used for this project contained a larger quantity of oddities such as incorrect grammar and spelling mistakes.

Another reason for the shortcomings of the LSTM and particularly the DistilBERT models, was a lack of time and computing resources, as there very likely are model designs that would perform better, but they were not found.

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4*(11), e00938. doi:https://doi.org/10.1016/j.heliyon.2018.e00938

Bais, R., Oder, P., & Ou, S. (2017). *Sentiment Classification on Steam Reviews.* Stanford University. http://cs229.stanford.edu/proj2017/final-reports/5244171.pdf

Clement, J. (2021). *Market size of the video games industry in the United States from 2010 to 2021.* Statista. https://www.statista.com/statistics/246892/value-of-the-video-game-market-in-the-us/

Devlin, J., Chang, M., Lee, K., & Toutanova., K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Google AI Language. http://arxiv.org/abs/1810.04805

Epic Games. (2021, January 28). *Epic Games Store 2020 Year in Review.*. https://www.epicgames.com/store/en-US/news/epic-games-store-2020-year-in-review

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Kiran, T. D. V., Reddy, K. G., & Gopal, J. (2017). Long short-term memory. *Journal of Chemical and Pharmaceutical Sciences, 10*(1), 175-178.

Muller, A. (2021). *Video Game Review Analysis.* url: https://github.com/MullerAC/video-game-review-analysis#readme

Möbius, (2021, December 14th). *Steam Game Review Dataset.* url: https://www.kaggle.com/arashnic/game-review-dataset

Pennington, J., Socher, R., & Manning., C.D. (2014). *GloVe: Global Vectors for Word Representation.* Computer Science Department, Stanford University.

Roy, A., Khan, M. H., & Chakraborty, S. (2018). *Analyzing Users' Sentiment towards Video Games Based on Reviews from Microblog.* BRAC University. http://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/10143/13301130%2c13201039%2c14101107_CSE.pdf?sequence=1&isAllowed=y

Sanh, V., Debut, L., Chaumond, J., & Wolf., T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* Hugging Face. http://arxiv.org/abs/1910.01108

Tang, P., Steinbach, M., Karpatne, A., & Kumar V. (2019). *Introduction to Data Mining* (2nd ed.). Pearson Education.