Binary sentiment analysis of video game reviews

Giuseppe Malara

28 September 2023

1 Introduction

The gaming industry has quickly become one of the biggest entertainment industries on the market, and can boast a revenue that surpasses the movie and the music industry combined (Shaw et al., 2020). From the rapidly growing development cost of video games (Zollner, 2023), and the closures of studios due to the release of consecutive unsuccessful games (Stewart, 2023), it can be argued that releasing a well-received game is paramount for gaming companies. In this context, reviews of previously released video games can be of great help to game developers as "sentiment analysis of videogame reviews can provide game designers as well as game researchers with insights on what the users/players perceive as favorable or unfavorable about a particular game." (Zagal et.al, 2012).

This project focuses on conducting binary sentiment analysis on user-submitted reviews of video games released on the gaming platform Steam, the leading video game distributor on PC. Reviews are categorized as either positive or negative, and an attempt is made at correctly classifying them based solely on the contents of the reviews through the use of different machine learning algorithms, namely Multinomial Naive Bayes (MNB) and Convoluted Neural Networks (CNN).

The dataset that was used in this project is the Steam Reviews Dataset by Antoni Sobkowicz (2017) and is publicly available on Kaggle. It contains over 6 million reviews in the English language published on Steam. Each row of the dataset contains the review text, the sentiment expressed within the review (either positive or negative), the associated game's ID, and whether there were users who found the review helpful. Handling these reviews made it evident that the features differ significantly from game reviews left by professional journalists, and the use of sarcasm and internet slang is much more present, rendering the sentiment analysis task even more arduous.

Results and findings point towards the CNN model performing better at this task, and will be reported in more detail in the Results section of the report.

2 Data Description

The data is provided in the form of a single comma-separated values (CSV) file with five columns:

- app_id: Number that represents the unique identifier of the game.
- app_name: Title of the game that is being reviewed.
- review_text: Text containing the review of the game.
- review_score: Binary value describing whether the review is positive (1) or negative (0).
- review_votes: Binary value describing whether the review was recommended by another user (1) or not (0).

While the first two columns could prove useful for analyzing a subset of reviews pertaining only to a certain game or for filtering out other games, the review_votes column proves to be a bit more problematic. The main limitation of this information is that it does not provide the exact number of users who rated the review as useful, but only an indicator that at least one person did. This means that a review that was rated as helpful by a single user has the same weight as a review that was deemed as such by 100.000 users, for example. Furthermore, "usefulness" is not clearly defined, and reviews that try to be funny or sarcastic, as examplified in Table 1, are considered "useful" even if they do not talk about the game at all.

app_id	app_name	review_text	review_score	review_votes
10	Counter-Strike	Ruined my life. 10/10	1	1

Table 1: Example row from the raw Steam Reviews Dataset

The dataset contains around 6.4 million reviews with 83% of the reviews being positive and 18% of the reviews being negative. The dataset has enough instances that it is possible to solve this imbalance by only taking into account a number of positive reviews equal to the total number of negative reviews. If one were to use all negative instances in this dataset, this method would create a dataset of around 2.3 million instances, still big enough to conduct meaningful operations or analysis, and to avoid the use of popular oversampling techniques such as SMOTE.

The data still remains noisy, and will therefore be filtered and pre-processed before being split into training, validation and testing partitions.

3 Filtering and pre-processing

3.1 Filtering

Since the Steam Reviews dataset contains more than 6.4 million reviews, computationally-light filtering procedures are taken before conducting heavier operations such as text pre-processing on a sample of the whole dataset. The first step is to drop duplicate and empty reviews. However, instances that do not specify which game they review are not discarded. This is because the models will only take into consideration the text of the reviews, regardless of which game they belong to. Besides, upon manual inspection, these reviews seemed to be valid.

Next, the dataset is filtered to contain a maximum of 5000 reviews of any given game. This is to avoid feeding the models too much data from a single game or genre, not allowing them to generalize on features common to other games. The number 5000 was chosen due to the fact that the final dataset will contain 300.000 instances, and 5000 represents an acceptable number of "similar" reviews, as it would be around 1.5% of the dataset.

During initial analysis it was noted that Steam censors swear words with a sequence of heart characters equal to the length of the swear word. This creates a problem because swear words, although it could be argued that they have negative valence by themselves, only seem to emphasize concepts and therefore prove difficult to replace with a regular expression or other methods. In this case, the choice was made to delete them entirely and rely on the rest of the review. Since hearts are removed during pre-processing, an additional step of removing reviews shorter than three tokens will be applied after pre-processing to account for reviews composed mainly of swear words.

Finally, a subset of 350.000 instances, 50% positive and 50% negative, is created before text pre-processing. This is a bit larger than the final 300.000 instances dataset that will be used to create the partitions. This measure is taken to account for empty or short instances created by some pre-processing steps.

3.2 Pre-processing

The reviews present in the dataset are noisy, and exhibit characteristics typical of internet slang such as the use of emoticons, presence of URLs, repetitions of characters etc.. These are handled through an extensive amount of pre-processing steps that aim to create a clean dataset. Stopword removal was considered and tested, but all models performed better without this step. This phenomenon is known to happen in datasets composed of web-crawled data, as shown in Hassan et al. (2014).

After pre-processing takes place, a sample of 300.000 instances is created and the training, validation and testing partitions are generated with a 70/15/15 split.

Simpler pre-processing steps will be described in the bullet list below, while more complex steps will be described in more detail in a separate sub-section. A complete list of all the pre-processing steps in the correct order can be found in Table 4.

- All the reviews are converted to lowercase.
- All URLs are replaced with the word "URL". The regular expression used to match them is ((www\.[\S]+)|(https?://[\S]+)).
- Sequences of 3 or more repeated letters in a word are replaced by one single instance of the letter.
- All punctuation characters are replaced with spaces. All sequences of spaces are then replaced with a single space.

3.2.1 Contractions

Due to their high frequency in the reviews, and in order to standardize the dataset and reduce the dimensionality of the document-term matrix, all contractions are expanded to their full form. This is achieved with the use of the *contractions* library. First, the contractions are identified by combining all the contraction patterns present in the contraction_mapping dictionary with a regular expression, then, the matched contraction is retrieved and replaced by its expanded form present in the dictionary.

3.2.2 Emoticons

Users frequently employ various emoticons in their reviews to express their satisfaction or dissatisfaction with the game. An attempt was made at replacing the most frequent emoticons that clearly convey a specific emotion. In practice, emoticons were grouped in a list by their emotion and replaced with a suitable word. During analysis it was also found that angle brackets were represented with their html tags "&lt" and "&gt". This was taken into account when writing the code.

Emoticon(s)	Emotion	Replacement
:(| **:c** | **:'(** | **:/**	Sadness or disappointment	sad
:) |**:D** | **:'D** | **:')** | **:P**	Happiness or satisfaction	happy
&lt3 (would be displayed as <3)	Love or approval	love
&gt (would be displayed as >)	Positive comparison	better than

Table 2: List of emoticons and their replacement

3.2.3 Emojis

Emojis are much more numerous and nuanced than emoticons, they can convey many more emotions and their meaning can change over time (Robertson et al., 2021). Attempting to identify their intended use and then replace them accordingly would be daunting and beyond the scope of this project, therefore, they are removed from the dataset through the use of the *unicode* library. Emojis are matched through unicode point ranges and then removed with a regular expression.

3.2.4 Ratings

Due to the dataset being a collection of reviews, there are numerous instances such as the one in Table 1, where the user gives the game a rating on a scale of 1 to 10 or 1 to 100 in the format of "80/100" or "11/10". This information could prove very useful in identifying the sentiment of the review, and will therefore be preserved in a different format.

First, a regular expression is made to match instances like "X/Y," where X and Y can be numbers between 0 and 100. Then, continuing to treat the rating as a fraction, a numeric value is extracted by performing the division, and is finally substituted by an appropriate word depending on the value. Table 3 shows the range of values and the substitute words.

Result of the "rating" fraction	Replacement
Less than 0.5	terrible
Less than 0.6	bad
Less than 0.8	good
Less than 1	great
1 or greater	excellent

Table 3: List of ratings and their replacement

1. URL replacement
2. Emoticon replacement
3. Ratings replacement
4. Emoji removal
5. Contractions expansion
6. Punctuation removal
7. Repeated letters replacement
8. Lowercasing

Table 4: List of pre-processing steps in order

4 Classifiers

Sentiment analysis is carried out on two different classifiers, Multinomial Naive Bayes and Convoluted Neural Networks. Both models were evaluated on the same testing partition. However, the MNB model was trained on a dataset that combined both the training and validation partitions, while the CNN model kept the validation partition to allow for parameter tuning. The following sub-paragraphs will describe the models in more detail.

4.1 Multinomial Naive Bayes

In this project, I employed the MultinomialNB classifier from the *scikit-learn* library's naive_bayes package, while leaving the Laplace smoothing parameter at its default value of 1. As for vectorization techniques, both CountVectorizer and TfidfVectorizer were tested, with the best results being achieved by TfidfVectorizer. Unigrams, bigrams and trigrams are included in feature representation, with a minimum document frequency of 5. Both presence and frequency features were tested with very similar results, but frequency features performed slightly better.

The best accuracy obtained is 0.874. Results are discussed in more detail in the Results section.

4.2 Convoluted Neural Network

In this project, two CNN models with different structures are tested. The implementation of these models is achieved through the *keras* library with the TensorFlow backend. The following procedures are taken for both models. The structure of each model will be discussed in its own sub-paragraph.

After loading the partitions, keras tokenizer is instantiated with a special token for unknown words (UNK). Afterwards, the tokenizer is fit on the training data to create a word-to-index mapping, which also includes a special token for padding (PAD). All the partitions are then converted to sequences of integers using the fitted tokenizer. To ensure that all sequences have the same length, they are padded or truncated at the end to a maximum length of 300.

After importing and loading fasttext's pre-trained word vectors, specifically 2 million word vectors trained on Common Crawl with 300 dimensions, an embedding matrix is constructed to store the word embeddings for all words in the word-to-index mapping. If a word is not found in the pre-trained embeddings, its corresponding row in the embedding matrix is filled with zeros. The sentiment labels in all partitions are encoded as integers using a label encoder with 1 for positive reviews and 0 for negative reviews.

After testing, it was found that both models performed best with the Adam optimizer set to a learning rate of 0.001, and binary cross-entropy as the loss function with label smoothing of 0.1.

To allow the models to generalize better and try to limit overfitting, two callbacks are employed during model training. The first is early stopping, which monitors the validation loss and stops training if it does not improve for three consecutive epochs. The best weights are restored upon stopping. The second callback is a learning rate scheduler that reduces the learning rate by a factor of 0.5 if the validation loss does not improve after one epoch.

The first layer is the same for both models, an embedding layer of size $(v+1)\times d$ where v is the size of the vocabulary (135104) and d is 300, which is the same as the embedding dimension of the pre-trained fasttext word vectors, whose previously generated embedding matrix is used for the initial weights.

Both models are trained for 10 epochs with a batch size of 16. Larger batch sizes were tested but resulted in worse performance, probably due to overfitting.

4.2.1 First CNN Model (2CONV1D)

The first model employs two CONV1D layers, the first layer has 256 filters, while 128 filters are used for the second layer. Both layers have a kernel_size of 3 with a relu activation function. Padding is set to valid since the data is already padded and truncated. Each convolutional layer is followed by a MaxPooling1D layer with a pool size of 2 to reduce dimensionality of the data and extract the most salient features, and a Dropout layer set to 0.5 to avoid overfitting. Finally, a Flatten layer is inserted before a Dense layer with 300 units and the final output layer.

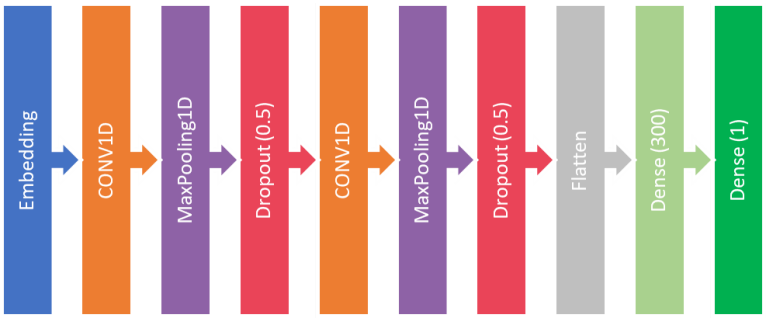

Figure 1: Structure of the first CNN

4.2.2 Second CNN Model (4CONV1D)

The second model employs 4 convolutional layers instead of 2. The CONV1D layers have a kernel_size of 5 while the filters progressively decrease from 600 to 300, 150 and finally 75. Max pooling layers are not used in this model, while one dropout layer set to 0.4 is inserted before the first convolutional layer and another one set to 0.5 is placed before the output layer. In this model the dense layer before the final output layer has 600 units. If not otherwise specified, parameters are the same as the first model.

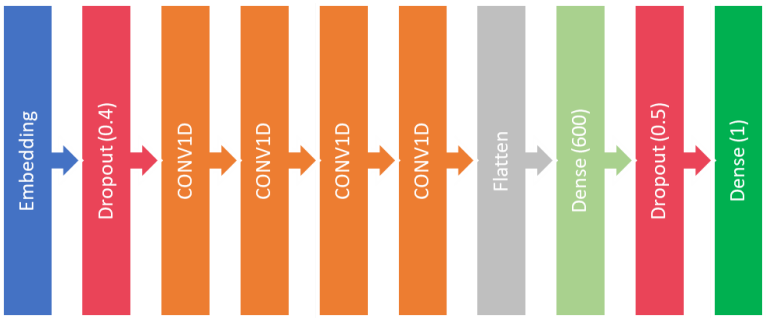

Figure 2: Structure of the second CNN

5 Results and discussion

The results of the binary sentiment analysis task indicate that the CNN models outperform the Naive Bayes classifier. Although the MNB classifier's performance is surprisingly good, achieving a test accuracy of 0.874, the CNN models demonstrate higher test accuracy scores, with the first CNN achieving 0.893 and the second CNN achieving 0.900.

From the table comparing the major metrics across all models, it can be seen that the MNB model has a higher recall and lower precision for the positive class, while the opposite is true for negative instances. In other words, it tends to classify ambiguous instances as negative reviews. One of the factors contributing to this phenomenon could be instances such as the one in Table 1, where a positive instance contains negative traits because the user reviewed the game in a funny or sarcastic manner. This occurrence is slightly mitigated in the 2CONV1D model, where the precision gap has reduced from 5 points to 3, and the recall gap has reduced from 7 to 5. The 4CONV1D model almost eliminates the gap, as there are only two points of difference in the precision and recall gap of both classes. Moreover, it improves every metric except for dropping one point in recall to the 2CONV1D model, and seems to be much more consistent.

Model	**MNB**		**2CONV1D**		**4CONV1D**	
Precision	POS: 0.91	NEG: 0.86	POS: 0.91	NEG: 0.88	POS: 0.91	NEG: 0.89
Recall	POS: 0.85	NEG: 0.92	POS: 0.87	NEG: 0.92	POS: 0.89	NEG: 0.91
F1-Score	POS: 0.88	NEG: 0.89	POS:0.89	POS:0.90	POS: 0.90	NEG: 0.90
Accuracy	0.874		0.893		0.900	

Table 5: Performance metrics of all classifiers

6 Limitations of the study and future research

In the process of constructing and evaluating the two CNN models, overfitting emerged as a primary concern. Despite efforts to mitigate this issue by employing MaxPooling1D and Dropout layers with substantial sizes, additional techniques could be employed to further reduce overfitting. For instance, adjusting the batch size and implementing L1 or L2 regularization may prove beneficial. Moreover, while the study explored various filter and kernel sizes, it is possible that more extensive hyperparameter tuning could yield improved performance.

Experiments were simultaneously being carried out on a dataset that contained all the negative instances in the dataset and an equal number of positive instances, for a total of 2.3 million instances. However, due to computing limitations, the final dataset used in this study is limited to 300.000 instances. Early experiments on the 2.3 million instances dataset seemed promising, therefore training and testing the 4CONV1D model on the larger dataset could perhaps lead to better results.

References

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A. (2018). Advances in Pre-Training Distributed Word Representations, Proceedings of the International Conference on Language Resources and Evaluation.

Robertson, A., Liza, F. F., Nguyen, D., McGillivray, B., & Hale, S. A. (2021). Semantic journeys: quantifying change in emoji meaning from 2012-2018. 4th International Workshop on Emoji Understanding and Applications in Social Media 2021.

Saif, H., Fernández, M., He, Y., & Alani, H. (2014). On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter. International Conference on Language Resources and Evaluation.

Shaw, S., Quattry, S., Whitt, J., & Chirino, J. (2020). Global gaming industry takes center stage. Morgan Stanley. Retrieved from: https://www.morganstanley.com/im/publication/insights/articles/article_globalgamingindustrytakescentrestage_en.pdf

Sobkowicz, A. (2017). Steam Review Dataset. .Zenodo. https://doi.org/10.5281/zenodo.1000885

Stewart, M. (2023). Saints Row developer Volition Games has been shut down. Game Informer. Retrieved from: https://www.gameinformer.com/2023/08/31/saints-row-developer-volition-games-has-been-shut-down

Zagal, Jose & Tomuro, Noriko & Shepitsen, Andriy. (2012). Natural Language Processing in Game Studies Research An Overview. Simulation & Gaming. 43. 356-373. https://doi.org/10.1177/1046878111422560

Zollner, A. (2023). Major publishers report AAA franchises can cost over a billion to make. IGN. Retrieved from: https://www.ign.com/articles/major-publishers-report-aaa-franchises-can-cost-over-a-billion-to-make