Predicting Game Recommendations from Reviews

Wesley Kwan, Aaron Chan, Michael Kusnadi
wekwan@ucsd.edu, ayc071@ucsd.edu, mkusnadi@ucsd.edu

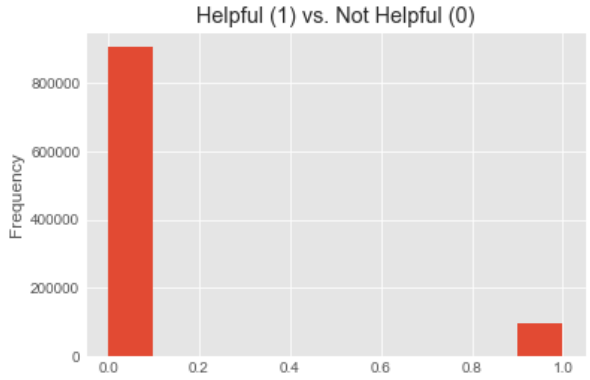
Abstract

As gaming becomes more and more prevalent in modern society, what makes or breaks a game is often not determined by any individual's personal opinion, but rather by how individuals perceive other user's opinions. Game reviews and recommendations have become a large factor in determining the success of a game before it even reaches the hands of some consumers and in this paper we look at review text along with recommendations.

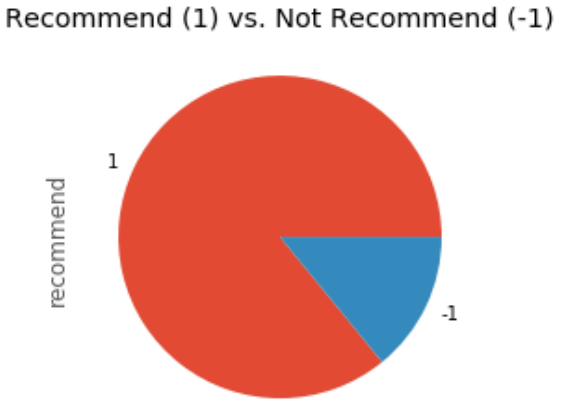
Using the dataset found on zenodo.org [1] , we aim to predict if a user has recommended a game based on corresponding review text the user wrote for that game using both traditional classification methods as well as some complex methods.

Data Exploration

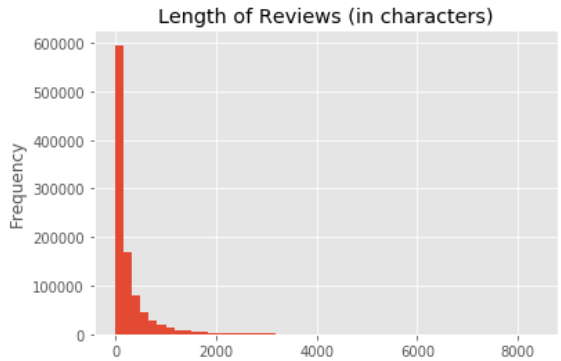
The dataset contains around 6.4 million Steam reviews. It has game ids that uniquely identify the game and review text given by a user pertaining to the game. The data also contains two binary features, recommendation, which denotes if a user recommends the game, and helpfulness, which denotes if the review was marked helpful on Steam. We will mostly be focusing on using the text data as features with the binary column recommendation as the label. As such, we remove the rows with NaN values for its review text. Though the original dataset has 6,417,106 rows, for the sake of time and computability, we only worked on the first 1,000,000 rows. A surface exploratory analysis on our first million rows shows: the number of unique games is 578, the number of unhelpful reviews is 903,697 (or 90.3697%), and the number of reviews where the user recommended the game was 858,878 (or 85.8878%).

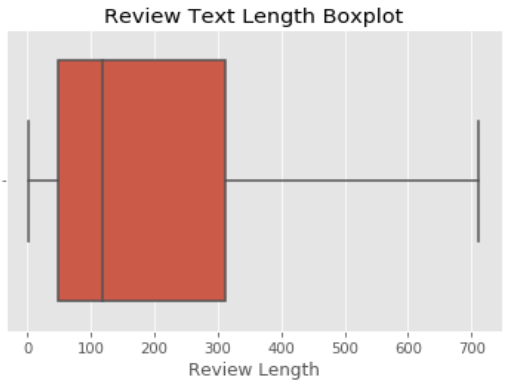


It is very important to note the large class imbalance that occurs with recommendations. If we were to naively predict "recommended," we would receive an accuracy of 86%.



Looking a bit at review text, we find that the average length of a review is about 300 characters (299.962 exactly) while the median is 119.





Objective and Expectations

Our objective with the dataset is to predict if a user recommends a game given features derived from the text of their review. We will evaluate our models using accuracy and an F1 score on "not recommended" reviews because of the very high class imbalance. For the baseline models, we will use bag of words feature vectors on a support vector machine and logistic regression.

For the baseline models, we first clean the review text by making all characters lowercase and removing all punctuation and extraneous white space. Using sklearn's CountVectorizer, we transformed the cleaned text into bag of words vectors. We then split the data into training, validation, and test sets and trained a support vector machine (LinearSVC) and logistic regressor, both using the regularization parameter C = 1.

We decided on two different approaches to our data. First, a traditional approach similar to the baseline models using TF-IDF feature vectors, then, a more advanced deep learning approach using recurrent neural networks. For the traditional approach, we will use TF-IDF, as using TF-IDF features will improve on the baseline score since the feature vectors would provide a richer representation of the input as compared to a bag of words and provide a better split between positive and negative reviews. We also believe that a recurrent neural network would work well due to having a large dataset and having a temporal element in language.

With the TF-IDF approach, we can use the same cleaned text for the baseline models and use sklearn's TfidfVectorizer to create TF-IDF feature vectors. For the recurrent neural network, we clean the text of non-alphanumeric characters, but keep capitalization and punctuation since they are a factor in the temporal nature of language. We use word embeddings, specifically the word2vec continuous bag of words variant. The feature vectors we feed into our network will look like the following: 0 for unknown words or the index of a word from a word to index dictionary. These vectors will be zero padded and fed into the embedding layer of our network, where it will learn word embeddings.

Designing Models

We use support vector machines and logistic regression as they are traditional classifiers. As explained earlier, TF-IDF representations should perform better than the baseline bag of words approach since it better identifies words important to each class.

Our other approach of using deep learning with a recurrent neural network (RNN) should also perform better as they work well with temporal data and language inherently has a temporal element to them. Deep networks, however, are susceptible to the vanishing gradient problem. Since gradients tend to get very small during backpropagation, this results in very little learning in the early layers of the network. Vanilla RNNs are known to be susceptible to the vanishing gradient problem since they compute the gradient through both layers and time. Because of this, they tend to not do well with long range dependencies. We will experiment with variants of RNNs including ones with long-short term memory (LSTM) units and gated recurrent units (GRU), both of which introduce gates that allow the gradient to flow backwards and more freely over time, helping to mitigate the vanishing gradient problem. We will also experiment with using bidirectional RNNs because the way that the English language is modeled: context from right to left can be as important as left to right.

Since neural networks have many hyperparameters to tune, we will discuss the ones that we found to be the most important. Before the network is made, we must choose an embedding space (100).. We then need to choose the length of the feature vectors that are fed into the network (200). The most important part is choosing the structure of the network. We decided to use an embedding layer, a RNN/LSTM/GRU layer, and a dense layer, to predict the class. Within the network, we used dropout to reduce overfitting and batch normalization to reduce overfitting and speed up training. Lastly, after some experimentation we chose the number of hidden units for the RNN/LSTM/GRU layer to be 128, as less units could not provide the needed complexity.

A large problem was scalability. Even with the smaller subset of data from the original dataset, we had trouble training the neural networks at this scale. Given the number of parameters that needed to be trained, our GPU's frequently crashed during training. To help alleviate the problem of scalability we trained in mini batches of size 256 on 3 epochs. Though neural networks are known to overfit, we were able to avoid this problem with the massive dataset and applying dropout in the network.

Comparing our models, we can see that the traditional approach using SVMs and logistic regression may be too simple, though they are relatively quick to train. Using the RNN approach is on the opposite side of the spectrum in that it is very complex, but is much more time consuming to train. Looking at the RNN models, they each have their own advantages and disadvantages as well. The vanilla RNN would not perform well due to the vanishing gradient problem. LSTM networks are a tried and true method, but have more parameters and are more complex than GRUs, making them harder to train. The bidirectional RNNs give a better model of language, but the bidirectional nature means that it has twice the number of parameters to train, taking up even more computing power.

Related Literature

There is prior research delving deeper into the effects that player reviews have on a game's performance indicators as well as research on how to apply these criticisms to improve their products. Livingston et al [2011] discussed the effects on negative and positive reviews on player experience [2]. Some very interesting points were how they found that players who had read a negative review prior to testing a specific game gave significantly lower ratings than players who had read a positive review prior to testing. They had also found that negative review text had a larger bias effect than positive review text did.

In a paper discussing factors in creating a successful game by Ahn et al [2017], Steam user reviews were used to find what factors played into creating a popular or unpopular game. They extracted features using the Kano model and performed semantic analysis to split words into what factors they correlate with the Bass diffusion model. They then found which words corresponded with popular ratings and unpopular ratings. A summary of their findings show that challenging, fairly priced games with a well structured story line are major factors in making a popular game based on user reviews.

Lin et al [2018] address the similarities and differences in PC game reviews as compared to mobile game reviews using Steam text reviews as their dataset for PC games. Some significant findings in similarities include: negative ratings are often posted very soon after playing, approximately half the time of hours played for positive reviews, most reviews for free to play games are made in the first hour of playing, and that players often complain more about game design than bugs in negative reviews.

These research delves much deeper into the effects and possibilities that user review data can accomplish and the implications that it can have on producers of all types of games.

Results and Conclusions

All RNNs were trained for 3 epochs.

Model	Accuracy	F1 Score (on not recommended)
Baseline: BoW SVM	0.91556	0.66
Baseline: BoW Logistic Regression	0.91350	0.62
TF-IDF SVM	0.92349	0.69
TF-IDF Logistic Regression	0.92066	0.67
Vanilla RNN	0.89443	0.56
RNN with LSTM	0.92419	0.70
RNN with GRU	0.92443	0.71
Bidirectional RNN	0.87407	0.50
Bidirectional RNN with LSTM	0.92433	0.72
Bidirectional RNN with GRU	0.92329	0.69

Looking at our results, we can observe many points we originally discussed. Unsurprisingly, the vanilla RNN and its bidirectional counterpart did not perform well, dropping below the baseline score. We can also see that RNNs with LSTM and GRU outperform the traditional approaches, but not by much surprisingly. With better computational power, we would expect that this should not be the case however. More layers, hidden units, and epochs should be able to boost the RNN significantly past the traditional approaches as RNNs can handle much more complexity. It is also worth noting that the bidirectional RNN with LSTM and GRU did not perform significantly better than the unidirectional ones. Some important takeaways from our results are: vanilla RNNs are not the best approach for our classification task, bidirectional RNNs do not seem to have a large edge on their unidirectional counterparts (in the case of the bidirectional RNN, it performs worse), and that traditional approaches can get very good results with very little training time and computation power.

References

[1] Antoni Sobkowicz. (2017). Steam Review Dataset (2017) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1000885

[2] Livingston, Ian & Nacke, Lennart & Mandryk, Regan. (2011). The Impact of Negative Game Reviews and User Comments on Player Experience. Proceedings - Sandbox 2011: ACM SIGGRAPH Video Game. 10.1145/2037692.2037697.

[3] Ahn, Sang ho & Kang, Juyoung & Park, Sang-Un. (2017). What makes the difference between popular games and unpopular games? Analysis of online game reviews from steam platform using word2vec and bass model. ICIC Express Letters. 11. 1729-1737. 10.24507/icicel.11.12.1729.

[4] Lin, Dayi & Bezemer, Cor-Paul & Zou, Ying & Hassan, Ahmed E.. (2018). An Empirical Study of Game Reviews on the Steam Platform. Empirical Software Engineering. 10.1007/s10664-018-9627-4.