STEAM Review Data Analysis

Myung Sub Cho, Zhanpeng Zhang, Dan Peterson
University of California – San Diego

Introduction

Today's leading businesses have already begun embracing natural language processing (NLP) as a key feature to improve efficiencies and gain competitive advantages in their industry. Developing a predictive classification model can help a business digest and gain understanding from large amounts of unstructured data and use that information to improve both their internal operations and their external offerings. There are multiple approaches to building a predictive classification model and lots of excitement around ground breaking unsupervised learning models. In this report, we compare a cutting edge language model against two traditional language models, and we hypothesize that the cutting edge model will outperform the traditional ones.

An extremely common use case of a predictive classifier model is for sentiment analysis. Here, a company can gain insight from online reviews, from tweets and posts on social media platforms, and survey responses from their customer base and use it as a temperature gauge to understand how they are performing relative to their competitors. The video game industry is a prime candidate for this business application as there is a large amount of data already available in the form of game reviews, news articles, and online discussion forums. The unstructured information available is already quite focused, making this a good candidate for training a predictive classification model to predict customer sentiment. During game development, the company can already have an idea of customer likes and expectations to use as a framework. They could also leverage the data to inform further development and updates to games based on customer feedback. They can also use this to get a general idea of where the industry is going or should be going, allowing them to react faster than their competition in fulfilling customer needs.

There are multiple approaches of varying complexity that are available to develop a good predictive classifier, both supervised and unsupervised learning models, and this report aims to explore a few of these approaches to compare and contrast methods. We will then assess their suitability for different classification use cases in the video game industry today.

1. Exploring the Data

Our team decided to use a "Steam Game Review Dataset[1]" from Kaggle to build our predictive models. This dataset includes three files:

1. train - a comma separated value (csv) file with 17,494 rows (one per review) and five fields:

[1] Möbius, April 2021, "Steam Game Review Dataset"

- ‘review_id’ - a unique value for each individual review
- ‘title’ - the title of game being reviewed
- ‘year’ - the year the review was written
- ‘user_review’ - the field containing the text review
- ‘user_suggestion’ - a binary field with a 1 if the user recommends the game and a 0 if the user does not recommend the game.

2. test - a csv file with 8,045 rows (one per review) and four fields:
   - ‘review_id’
   - ‘title’
   - ‘year’
   - ‘user_review’

3. game_overview - a csv file with summary information held in five fields for 64 games:
   - ‘title’
   - ‘developer’
   - ‘publisher’
   - ‘tags’ - descriptive tags associated with each game
   - ‘overview’ - a summary of the game written by the publisher

As this dataset originates from a Kaggle competition, we do not have access to the ‘user_suggestion’ label for the test set and, therefore, we will use 20% of the training data with the same random seed for the split as our test set to assess performance. The difficulty of predicting users’ sentiments in this task is that Steam users often make sarcastic comments, which humans can easily spot but are hard for computer programs to detect.

Conducting some exploratory data analysis reveals insights into how the games were rated in the dataset:

Figure 1: Counts of Recommendations

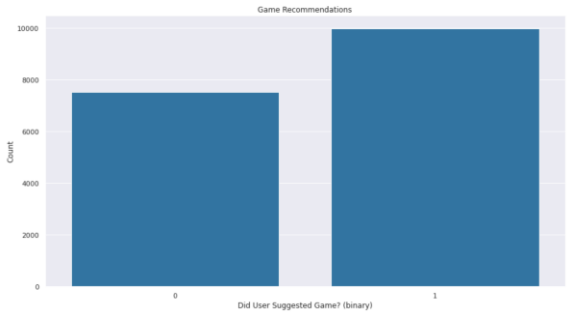


FIgure 2: Proportion of Recommendations by Game

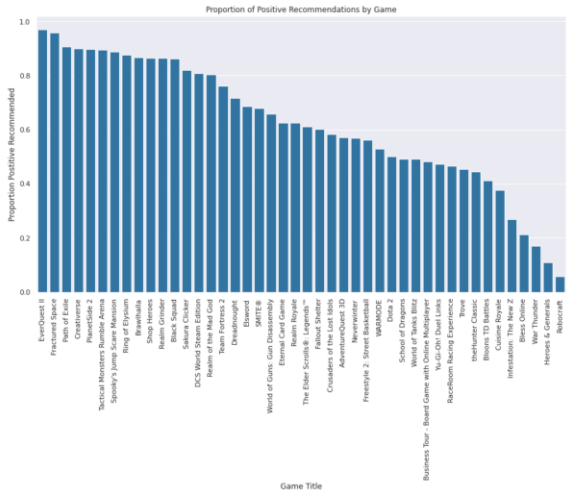


Table 1: Top 10 Recommendations (proportionally)

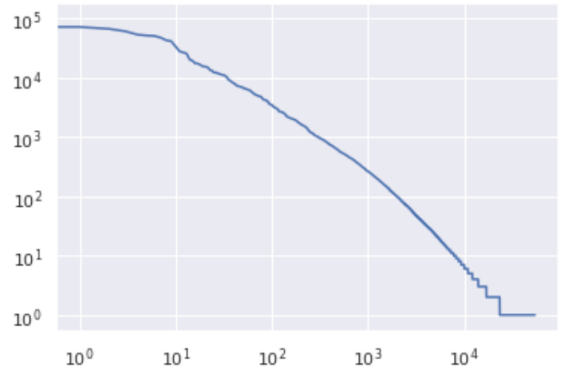
Rank	Title	Prop Rec
1	EverQuest II	0.971
2	Fractured Space	0.958
3	Path of Exile	0.906
4	Creativerse	0.900
5	PlanetSide 2	0.896
6	Tactical Monsters Rumble Arena	0.895
7	Spooky's Jump Scare Mansion	0.887
8	Ring of Elysium	0.876
9	Brawlhalla	0.866
10	Shop Heroes	0.865

Table 2: Bottom 10 Recommendations (proportionally)

Rank	Title	Prop Rec
35	RaceRoom Racing Experience	0.466
36	Trove	0.453
37	theHunter Classic	0.444
38	Bloons TD Battles	0.412
39	Cuisine Royale	0.376
40	Infestation: The New Z	0.269
41	Bless Online	0.212
42	War Thunder	0.169
43	Heroes & Generals	0.110
44	Robocraft	0.057

Finally, we looked at our corpus in terms of Zipf's Law - chiefly, that we can observe an inverse relationship between the rank of a word and the frequency that it is used in the corpus. That is, that the second most common word will appear half as often as the most common word, the third most common word a third as often, and so on[2]. This relationship is shown below by the seemingly linear line on a log-log graph:

Figure 3: Zipf's Law on our Corpus



2. Description of the Models

Our team approached this project using three classification prediction models: the first to determine a baseline for performance with Term Frequency - Inverse Document Frequency (TF-IDF) as our baseline and Skip-gram as a comparison. For our comparison approach, we decided on using Google's Bidirectional Encoder Representations from Transformers (BERT) model as this represents the cutting edge method in natural language processing.

Model 1: TF-IDF

The term frequency of a word is the number of times that word appears in a document. Document frequency measures how many documents in the corpus have a given word:

w = word, d = document, D = corpus

Term Frequency (tf):

$$tf(w,d) = |\{ t \in d \mid t = w \}|$$

Document Frequency (df):

$$df = |\{ t \in d \mid t = w \}|$$

TF-IDF:

$$tf\text{-}idf(w,d,D) = tf(w,d) \times \log_2 \left(\frac{|D|}{1 + df(w,D)} \right)$$

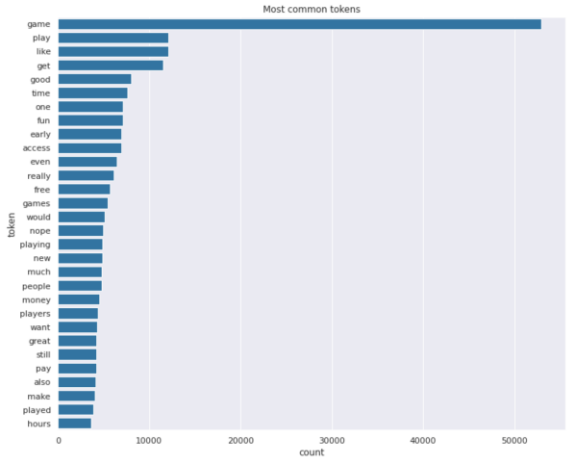
"For a word to be 'relevant' in a particular document, we want the term frequency to be high, and the frequency of the word across the entire corpus to be relatively low.[3]" This will pull out words that differentiate the word used in the document from the other words in our dictionary. This provides a weighted vector of features instead of just counts of words The TF-IDF approach takes the importance of certain words to certain documents into account, thus being able to specify the topic of a document. Our dictionary for this corpus of documents

[2] https://en.wikipedia.org/wiki/Zipf%27s_law

[3] Julian McAuley, *Personalized Machine Learning*, p. 229

contains 54,709 different words. After removing stop words and punctuations, and converting all the words to lowercase, this number drops to 48,198. A frequency of the top 30 occurring words after stopwords have been removed is displayed below:

Figure 4: Top 30 Words (minus stopwords)



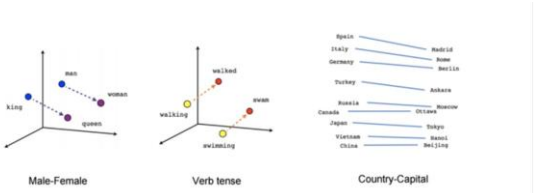
We then use a logistic regression to determine if the review is likely to be a positive recommendation, returning a 1, or a negative recommendation, returning a 0.

Model 2: Skip-gram

The drawback of the TF-IDF approach is that it doesn't consider the context of a word. Skip-gram, a type of word embedding model, addresses this problem by taking the surrounding words into account. Word embedding models take the words in each document, or user review, and featurizes them into a vector of numbers. The simplest and most common way to do this is through the Skip-gram model. "Both are architectures to learn the underlying word representations for each word by using neural networks. In the CBow model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. While in the Skip-gram model, the distributed representation of the input word is used to predict the context.[4]."

The idea of this Skip-gram model is to go through each document in the training corpus, and build a word representation for each word given a sliding window with the size of 15 that gives it a context. To reduce this number, we remove words from the dictionary if they occur less than 3 times. For each word in the dictionary, the Skip-model discomposes it into 300 dimensions by training a neural network with 300 neurons in the hidden layer, with a decimal number to represent its extent in each dimension. And for each document, we take the average of the word embeddings of all of the words in it, so that each document is the average of the words that it consists of. This gives us a vector 300 features long for each document. And on top of that, in order to handle the words that were unseen in the training corpus, we applied the pre-trained Fasttext model from the gensim library. Now that the word embeddings are built, a simple logistic regression model is trained to predict if the game is recommended, returning a 1, or not recommended, returning a 0.

Figure 5: Skip-gram General Concept[5]



Model 3: BERT

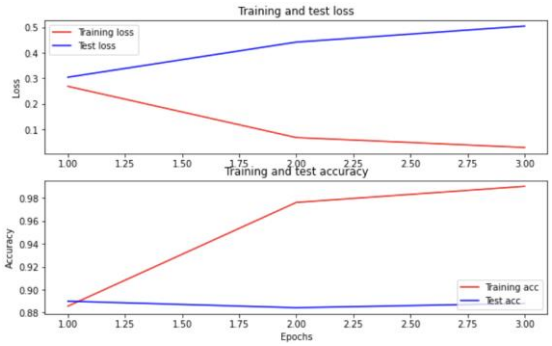
Our final model is based on the state-of-the-art deep neural net language model instead

[4] Ria Kulshrestha, *NLP 101: Word2Vec — Skip-gram and CBOW*

[5] Ankur Tomar, A math-first explanation of Word2Vec

of the previous traditional language models. What makes BERT different is that "BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus[6]". BERT is developed by Google that "randomly masks some of the tokens from the input… (with) the objective… to predict the original vocabulary id of the masked word based only on its context.[7]" This model is lauded for its superior performance in many different classic NLP tasks, including sentiment analysis and classification, and prompted our team to compare it to the word embedding model and the TF-IDF approach that we are more familiar with. Unlike word embedding models, BERT considers the order of words in a sentence. Moreover, it goes over the same sentence bi-directionaly and randomly masks some of the words, thus being able to capture much more information than other models. It also splits words into subwords, which enables it to understand the natural language more comprehensively, and even be able to understand words that it has never seen in the training corpus.

Figure 6: BERT Model performance



We implemented BERT with the pre-trained "bert-base-uncased" BERT model under TensorFlow. This BERT model is a smaller version of BERT, and it ignores cases just like our previous models. We found that the model becomes more overfit as they continue to train. There is almost no change in accuracy, but we concluded that epoch = 1 is the most suitable because the loss increases as the epoch increases. This overfitting seems to have occurred because the data is small data with less than 20,000 observations.

3. Assessing Model Performance

Since the labels are balanced as shown in Figure 1 and it only has two levels, it's safe to assess the performances of the models with accuracy. Table 3 below shows the contrast of the accuracies of the three models on the test/validation set. In assisting with accuracy, we also measure the models with F1-score Binary and ROC-AUC score so that we can assess the models properly

[6] Jacob Devlin and Ming-Wei Chang, Research Scientists, Google AI Language, *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing*

[7] Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT 2019, p.1-2

even if the labels are unbalanced, as shown in Table 4 and Table 5.

Table 3: Model Performance - Accuracy

Model	Accuracy
TF-IDF	82.5%
Skip-gram	85.7%
BERT	88.6%

Table 4: Model Performance - F1 Score

Model	F1-score Binary
TF-IDF	84.9%
Skip-gram	87.3%
BERT	90.0%

Table 5: Model Performance - ROC AUC

Model	ROC-AUC Score
TF-IDF	82.1%
Skip-gram	85.0%
BERT	88.4%

TF-IDF, as the most trivial model, gives an accuracy of 82.5%. Skip-gram, which takes contexts into account, gives an accuracy of 85.7%. And the state-of-the-art model BERT expectedly shows the best performance, with an accuracy of 89%. This proves our hypothesis that the cutting edge language model outperforms the two traditional language models in the classification task.

4. Sentiment Analysis Application

After building the models, we developed an interactive application that predicts whether the user recommends a game or not by taking in the user's input. Although the BERT model has the best performance in classification, it was built on Google Colaboratory which makes it difficult to develop a user interface. Thus, we developed the application with the Skip-gram model which doesn't give the best predictions. The URL to the application is here[8]. After downloading the zip file, the user shall unzip it and run the "final-project.exe" file. A command line windows will pop up first as shown in Figure 7, and the user shall wait for several seconds before the user interface shows up as shown in Figure 8.

Figure 7: Command Line Screen

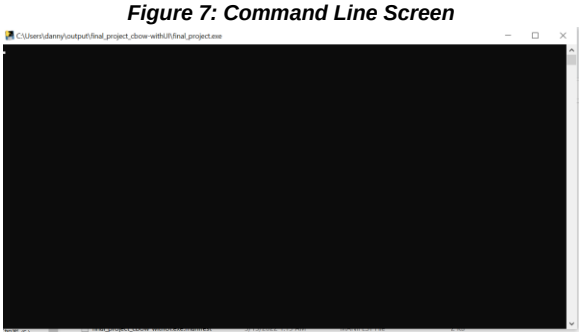
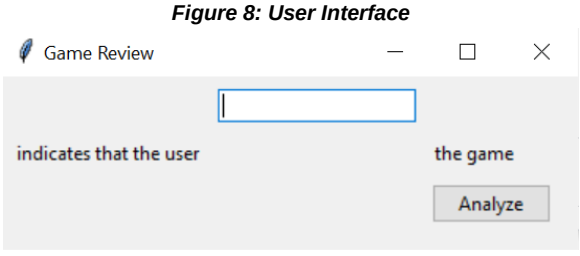


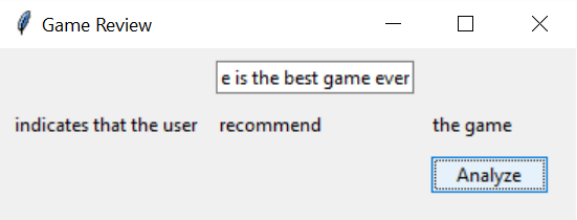
Figure 8: User Interface



8 https://drive.google.com/file/d/19wCEEXQqXU5pDIgM0d3xYFvlWgiBAtqx/view?usp=sharing

The user shall enter a game review with English words in the input box, and click the “Analyze” button to get the prediction as shown in Figure 8. The application will show either “recommend” or “doesn’t recommend” depending on the prediction of the model. Note that the prediction result will not change when the application is not able to give predictions.

Figure 8: User Interface Output



5. Conclusions

The video game industry is a prime candidate for this business application as there is a large amount of data already available in the form of game reviews, news articles, and online discussion forums. An improved and modified version of this program could prove critical to a video game company’s understanding of its user base and help with strategic decision making.

Before, during, and after a game is created and released the company can make informed decisions about their next business move based on customer discussion and feedback. They can use this as an advantage to react faster and more accurately than their competitors, creating value for both the company and their customers.

Most video game players (customers) post their reviews on social platforms like Twitter and Reddit. Unlike the Steam review data that shows "whether the user recommends it or not," there is no indicator of whether the user likes it or not. Our trained model can be used to judge whether the customers like or not the product. This will provide good business information not only to the game maker but also to publishers and investors.

In the end, the cutting edge BERT model outperformed the TF-IDF Bag of Words model and Word2Vec Skip-gram model, making it a more suitable predictive classification model. The BERT model does require more computational power and time than the other models, but is less of a problem for businesses with high spec machines. However, considering the overfitting problem shown in the model training process, it is necessary to collect more observations.