***Abstract*** – Video games are rapidly becoming one of the most accessible medium for entertainment worldwide. Its rise coincides with the emergence of unstructured data – of which includes textual and other multimedia data - as a valuable source for insight, and with advancements in data science it has become increasingly more feasible to process these data into a digestible, meaningful form. Using methodologies in NLP analysis, this project aims to process user reviews to output insights and create a predictive model to anticipate recommendation status.

***Keywords*** – NLP, Review, Sentiment Analysis, Topic Modeling

1. ***INTRODUCTION***

With the rise of the digital age, video games have become one of the most popular and prominent entertainment mediums for people globally. Though many factors contribute to its emergence, the advent of publicly accessible online marketplaces have made video games accessible to anyone with at least an internet connection and a computer. These platforms possess a wide array of features to its users – but one of the most critical to a game’s success is the review sections.

While individual customers can easily make use of reviews for their own benefit, game developers often face the challenge of parsing reviews to extract insights from these texts to help tackle business problems. Thus, this project strives to analyze text reviews and translate them into tangible insights, using Natural Language Processing techniques to help solve common business problems related to game development. These may include several of the following questions:

* What is the general sentiment on the game?
* How do we anticipate user reviews?
* What are the issues the users are facing?
* How are the game systems received by users?

1. ***METHODS***

For the purposes of this research, we will be utilizing a publicly available dataset from Kaggle. The dataset includes various metrics as retrieved from the Steam marketplace, and lists several of the most popular games available in 2018 of which included user votes (on helpfulness/funny), review date, user recommendation and the actual review corpus itself.

We will be making use of a few NLP techniques:

* Sentiment Analysis
* Topic Modeling

In addition, to tokenize text, analyze text and build the classifier we will be using the following:

* TF-IDF vectorizing method
* Multinomial Bayes Classifier

1. ***ALGORITHMS***

* **TF-IDF Vectorize**

TF-IDF (Term-Frequency-Inverse Document Frequency) is a vectorizing method that helps identify a word’s relevancy in each text by calculating its frequency in a given corpus. The method is split into several of the following parts:

**Term frequency** (tf): In a given document, term frequency is the number of occurrences of a word. Since order does not matter, we can use a vector in a bag-of-words way. The higher the frequency, the more significant a given word W is to the document d.

**Document frequency** (df): Similar to term frequency, with the key difference being that document frequency describes the number of occurrences in the document set; for example, when analyzing food review texts, document frequency is the frequency of a word W across all review documents in set N. We can describe this as follows:

**Inverse document frequency** (idf): This tests for word relevancy and computing the weightage by considering tf and df. To compute idf, we can begin by assuming that in a given set of documents N and a word W:

Putting all of these together, we can calculate the weightage for each word W as follows:

* **Multinomial Bayes Classifier**

The Multinomial Bayes model is a variant of the Naïve Bayes model that specifically works well with text documents. A Naïve Bayes model is an algorithm that bases itself upon conditional probability and assumes that within a dataset, all its features are mutually independent from each other. That is, the occurrence of one feature does not influence the probability of occurrence of another feature.

The Multinomial Bayes model is an event model used to specifically process discrete features (i.e. word counts). It makes use of the *Bag of Words* model to tackle text data – typically derived from large documents characterized by high dimensionality and numerous features. The model treats documents as if they are a bag of words, assigning each individual words as features with each corpus as a random assortment of them.

1. ***ANALYSIS***
2. *Exploratory Analysis*

To begin, we will first perform exploration on the dataset, closely inspecting features that we are most interested in. For nomenclature’s sake, in this analysis we will consider “recommended” status as a positive review while “not recommended” is a negative review.

The dataset possesses several features that may be of interest. The date posted column describes the date when the user posted the review on Steam. In this dataset, the data starts from 2010 and tracks only partway through Q1 2019 mid-February.

The dataset also has features called “funny” and “helpful”, which allows users to vote on whether a review has the trait. The data also distinguishes if the user has had early access to a game and based the review on it; for this specific dataset, however, none of the reviews have been early access reviews.

The real crux of the project is in the review column, which contains the review text as inputted by the users. Typically, these reviews are unstructured and come in many forms like one-liners, ASCII art, or long-form prose, and can also come in different languages. This dataset has extracted specifically English language reviews, and has

Furthermore, the dataset provides a variety of different games of various genres and sizes. For the purposes of this research, we would look at the game “Grand Theft Auto V” (GTAV), a game with a relatively large review count. Some of the game’s most important features according to its Steam page are as follows:

* High graphical fidelity
* Engaging story and real world-inspired setting
* Online multiplayer
* Expansive modding tools

1. *Visualizing Data*

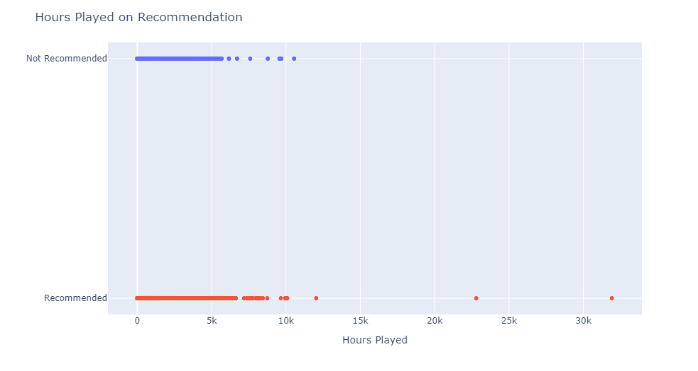
In figure 1, we illustrate the distribution of review lengths, done by creating the new variable by counting the strings in each review. As can be seen here, a large majority of reviews are skewed towards shorter reviews than longer ones – perhaps indicating that the vocabulary is more limited. This is also more in-line with the general assumption that reviews don’t tend to be overly lengthy.

Timeline

Description automatically generated with low confidence

*Figure 1: Distribution of review lengths*

Next, we illustrate the variance in hours played for each users who left reviews to potentially identify any clear user patterns before actually processing the text. As indicated, there is a slight difference in hours played among users who recommended against users who did not.



*Figure 2: Distribution of review length on hours played by recommendation status*

Lastly, we depict the counts for the recommendation categories. This gives a cursory glimpse of the general user sentiment towards the game and can give context to the model we will be building later. Figure 3 indicates that while there are more positive reviews than negative, the size of negative reviews is large enough to consider the game a polarizing game. Perhaps there are elements in the game that are controversial, which does indicate that further exploration might elaborate more on what the general sentiment conveys.



*Figure 3: Recommendation counts*

1. *Text Preprocessing and Classifying*

Before we can build the classifier, there are several measures we must take to ensure results make sense. In NLP analysis, one of the key steps is to deal with redundancies and non-alphanumeric characters.

Redundancies in the corpus are represented by extended forms of words. For example, in a given dataset we can extract the word “preparedness” and “prepared”, two words that likely have the same meaning but will show up as different features in the output. We can remove these suffixes on words thus performing “stemming”. This procedure is not entirely reliable especially if removing the suffix from the word does not output a meaningful word (i.e. “laziness” becomes “lazi”). Instead, we can perform “lemmatization” and convert these words back to their root form.

The other process is to filter stop words, which are filler words used to form complete sentences but are otherwise useless to us in our model.

The *spaCy* package stores all stop words in a class, which we can store in a variable to filter out. Additionally, we can add more words into this variable as required depending on the output.

1. *Sentiment Analysis*

The first part of the project is to conduct a prediction using the text data. This will involve the use of the classifier and the vectorizer. Afterwards, from the scikit-learn package, we call the Multinomial Naïve Bayes classifier and the TF-IDF vectorizer.

We will leave our classifier unmodified. For our vectorizer, we will set the following parameters:

**Minimum *df***: We will be setting the minimum document frequency of a word to 5.

**Normalization**: We will be using the “L2” normalization method. This method scales the values such that when squared and summed, they equal to 1.

We will fit and transform our review text using the vectorizer, converting it into array form. Then, we will conduct a standard partition of 60-40 train-test. Then, we fit the training data into our model and predict values using the test data. For evaluation, we generate a classification report and a confusion matrix to highlight model accuracy.

1. *Topic Modeling*

The second part of the project is to depict the terms within the review as clusters. Immediately, we can utilize a simple Word Cloud to depict the document. When generating the word cloud, we can filter out the stop words. The cloud looks as follows:

Text

Description automatically generated

*Figure 5: General Word Cloud*

Further expanding on this, we can apply vectorization to the text and reduce dimensionality via the TruncatedSVD (TSVD) method, which follows PCA but does not center the data before computing the SVD value. Since our resultant matrix after can be mostly sparse, we must resort to this in place of PCA. We will set our desired number of components to 5, resulting in 5 clusters of words. Additionally, we set the maximum number of words in the cloud to 50.

1. ***RESULTS***
2. *Sentiment Analysis Prediction Results*

After initializing the model, we run the prediction and compare its results to the test set. The results indicate a high accuracy score, with high precision and recall.

Accuracy Score: 0.855

Chart

Description automatically generated

Figure 6: Classification Report

The figure below depicts the classification matrix from the above model. The type 1 and type 2 errors from the model is 3634 and 2162 respectively.

Chart, treemap chart

Description automatically generated

Figure 7: Confusion Matrix

1. *Topic Modeling Word Cloud Results*

For topic modeling, we will be clustering the words in the documents and displaying them in the form of word clouds. To start, we will call a new variable storing our TSVD function.

We will then define two functions – one to generate topics from our vectorized data and the other two generate the corresponding word clouds. We can use the spaCy package to filter out stop words through the word cloud. Finally, we can generate the word clouds. However, the search can be more specific; rather than looking at all review texts, we can conduct this analysis on only positive or negative reviews. The generated word clouds will be appended at the end of the document for brevity’s sake.

Word cloud 1 depicts the clusters generated from the text in general. From it, we can extrapolate several overhanging topics:

* Online experience
* Graphics
* World
* Mods

Word cloud 2 depicts clusters generated from positive reviews. Topics and insights that have emerged here include:

* Online experience
* World
* Generally positive sentiment
* Graphics

Some interesting terms tied to the general sentiment include ‘free’, which may be related to the ‘free-play’ weekends that allow users who don’t own the game to play it or discounts on Steam.

Word cloud 3 presents some interesting topics:

* Online experience
* Modding
* Single-player experience
* Rockstar (the developers of GTA V)

We can see that the user sentiment towards the game’s world and graphics are positive, and among reviewers who input positive reviews, all seem to enjoy the product overall. However, from the negative review cloud we can identify glaring issues as conveyed.

First, we can consider the game’s online experience as a polarizing feature since it has been brought up in both positive and negative reviews alike. The term is clustered around words like ‘ban’ and ‘illegal’ – perhaps conveying common issues prevalent in online games such as harassment, hacking and others. One possibility is that the users may be annoyed at the lack of action taken to address other users that are banworthy or otherwise detract from the user agreement.

Another polarizing term – ‘mods’ – is related to the act of modifying the game’s files to change certain assets that work in-game. This was also clustered with the words “cease” and “desist”, which is a legal action companies may take to any individual or business that infringes on their intellectual property. Brief research into this feature reveals that this may be tied to a popular modding tool for GTAV called “OpenIV” that received a cease and desist from the developers themselves, which may be a reason for the negative reviews. These terms are also clustered with “rockstar”, who are the developers of the game.

1. ***CONCLUSION***

Using a prediction model, game developers can anticipate user sentiment using existing reviews from platforms like Steam at a relatively high accuracy. There is room for further improvements – further lemmatization and filtering of stop words or use of alternative vectorizing methods may provide a more realistic output.

In turn, topic modeling is possible to cluster reviews divided into topics. The model generated here could identify several topics critical to the game as mentioned in its Steam page – graphics, online and mods - as well as criticisms of the game features and actions taken by the business outside of the game. Alternative options would be to choose different number of components for dimension reduction or further lemmatization or filtering.

Ultimately, these are powerful tools that are open-source and easily accessible to game developers that could process unstructured text into a more digestible format, and support game developers in directing their efforts on game development both pre- and post-release.

1. ***REFERENCES***
2. <https://www.kaggle.com/datasets/luthfim/steam-reviews-dataset?resource=download>
3. Grand Theft Auto V on Steam, <https://store.steampowered.com/app/271590/Grand_Theft_Auto_V/>
4. “TF-IDF Vectorizer scikit-learn”, by Mukesh Chaudhary, <https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>
5. GTA modding tool OpenIV shuts down due to cease and desist from Take-Two, by Christopher Livingston, <https://www.pcgamer.com/gta-modding-tool-openiv-shuts-down-claiming-cease-and-desist-from-take-two/>
6. <https://pypi.org/project/spacy/>
7. <https://amueller.github.io/word_cloud/index.html>

**Word Cloud Figures**

Word Cloud 1: Topic Model for entire data

Text, letter

Description automatically generated

Word Cloud 2: Topic Model for Positive Reviews

Text

Description automatically generated

Word Cloud 3: Topic Model for Negative Reviews

Text

Description automatically generated