Steam Reviews Summary Generator

Kelsey Dwyer
Bao Le

December 11, 2024

Abstract

Steam, an online video game distribution service, accumulates a massive amount of consumer review data where users express their thoughts and feelings regarding products. The multitudinous amount of textual data serves to be quite challenging for a user to review in its entirety. Many users who seek to learn more about popular games are faced with an abundant amount of reviews that can be overwhelming to sort through. In our paper, we sought to explore ways in which we could help condense these reviews in a digestible, summarized manner that serves to benefit both the user and Steam officials. Efficient filtering and word-pair summarization of the review data would allow users to gather a general consensus on a particular game quickly, which appeals greatly to user satisfaction. In addition, the generated main idea of word pairs summary will allow for a more pointed course of action for Steam officials particularly, in product promotion, product availability, and improving product performance. All of those strategies will work to boost overall revenue. The key aspects of our work centered around filtering the data to detect the most popular products based on the number of reviews it has received. From there, we were concerned with generating a summary for each game using Sentiment Analysis, Natural Language Toolkit, Pair Words, Cosine Similarity and Euclidean Distance.

1 Introduction

In our current technological era, many companies are surveying their consumers through online reviews of their products. With the growing population of internet users, companies are seeing a greater influx of reviews from their users. One of those companies is Steam. Steam is an online video game distribution service. It currently stands as the largest PC game distribution platform. If you are interested in learning about all the features Steam has to offer, visit: Steam website. Steam collects reviews from its users where they can express their thoughts on the platform and its products. Steam currently has over 100,000 games to offer with over two million users. With such a diverse collection of games and a multitudinous amount of users, the reviews will yield quite a diverse collection of data. Although Steam already has an overall rating system, it is hard for developers and customers to knows any more detail beside the distribution of likes and dislikes. We seek to identify these hidden recurrent themes in the review data and condense them through word-pair summarization. This will lead to a compressed presentation of the reviews that represent the sentiments expressed within the data allowing for efficient exploration of the reviews for both the user and Steam officials.

1.1 Related Work

Before we detail our process, it is important to identify similar work in the area of mining product review datasets that our project will join alongside.

In working with the Steam Review database, Wan and McAuley were interested in identifying monotonic behavior chains that explain the relationship between implicit and explicit signals in the

data [8]. Our project focuses primarily on explicit signals of the Steam Review data to help with the summarization. Other work done with automatic summary generation of game reviews can be seen through Yauris's double propagation technique for extracting pairs of aspects and its sentiment, then aggregating them by aspect categorization and presenting them as a summary [9]. In addition, Panagiotopoulos applied k-means clustering to a set of game reviews to identify groups of similar sentences, then, they applied sentiment analysis to each cluster in order to produce their summaries [5]. Lastly, the work done by Kosmopoulus addresses the challenging feasibility of the evaluation of game review summaries. Their work outlines a protocol that assumes that user study is one of the best courses of evaluation and sets a benchmark for how to conduct user study successfully. They establish an evaluation procedure protocol that emphasizes that a user's language fluency and gaming experience be assessed prior to their evaluation of the summaries. In addition, they had the best success when the user evaluation they conducted did not require a long duration of time for the user as fatigue was a contributor to noise in the participant's evaluation [2].

We also reviewed some more general systematic ideas for summarization tactics such as Text Mining. Text mining is useful as it allows for efficient detection of useful text while eliminating noise. This allows for a reduced data set of key elements that allows for easier detection of patterns and tendencies. Wen-Jie Ye and Anthony Lee discuss the use of text mining of featured words related to consumer sentiments in consumer reviews as a way of producing sentiment tendency graphs and summaries[10]. Filtering is another technique often applied in many avenues when working with complex datasets. Filtering is a tactic that works to narrow data into a more concentrated grouping based on some specified related variables. One filtering framework is identified by Landry's approach for analyzing higher-order data sets by filtering interactions by their size [3]. Visually seeing the degree to which input variables interact could lead to more precise conclusions. Similarity is another tactic that works with lists of words and combines similar words and synonyms with singular words or phrases. One recent Similarity technique described by Zhao's *An Improved K-Means Algorithm Based on Contour Similarity* [11] is the improved version of the traditional k-value algorithm from the 1960s. The function intention is to determine the similarity of articles or words by using Euclidean distance calculation. In addition, our work builds upon the work done with Global Vectors. Pennington's work focuses on creating word embeddings that account for both the syntax and semantics of words found in consumer reviews [6]. Thakur's work on extractive Text Summarization finds vector representations (word embeddings) for each sentence using GloVe embedding. From there she used TextRank to create a cosine similarity matrix which is intended to store the similarity scores of every sentence with one another [7] Lastly, other work in product review summarization has incorporated unique uses of clustering. Ly's work uses clustering to generate a summary of product reviews that include both user sentiments and justifications for those opinions [4].

After reviewing previous works, we constructed a plan by identifying many algorithms that would prove to be helpful in the product review summarization of our dataset such as Filtering, Sentiment Analysis, Pair Words, Cosine Similarity, Euclidean Distance and algorithms from Natural Language Toolkit (nltk). We will discuss our own process in the Methods section below.

Our Approach

2 Methods: Types of Algorithms

Dataset: We use data from the Steam review distribution network. [8]. We began by reading the raw data into Python and then saving the game ID and the review list as a dictionary (called gamelist) for each game.

2.1 Set the scope for data with Filtering

The Steam dataset doesn't have a filter mode. This means it has every user's review that it comes by no matter if they are from the same game or not, which results in a large diversity in the game's titles present in the data. However, Steam already has an overall scoring system that shows on the product's page whether that product has more likes or dislikes in the review section. Besides, any amount of reviews less than 10000 would be putting bias in a machine model for the game, while being manageable with enough people. We are focusing on games with a threshold of more than 10,000 reviews, which means we need to filter out unqualified games. To begin filtering our Steam Review dataset, we needed to gather an initial distribution of the dataset as a whole. Figure 1 below details how many reviews have been created for each game.

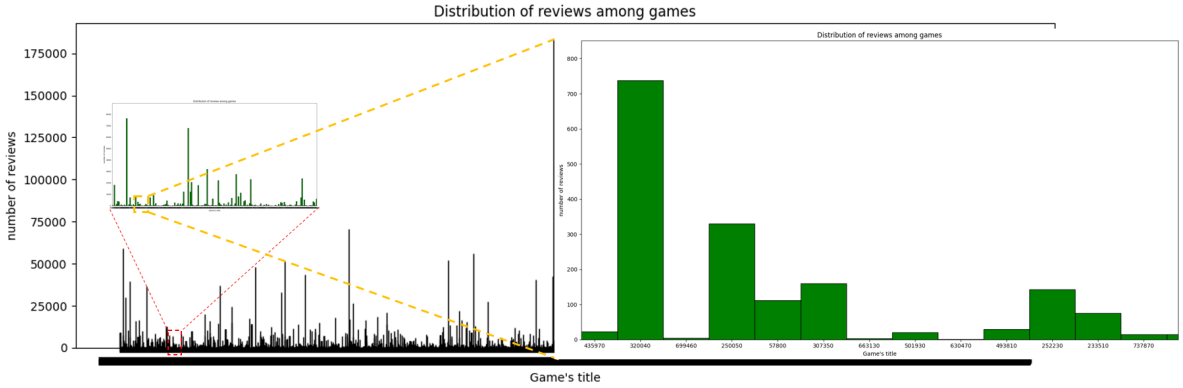

Figure 1: Distribution of Steam Reviews Among Games

This summary analysis is an important first step before filtering the data. It allows us to visualize what we have in total, so we can determine the best way to filter out data that does not serve our end goals. This figure also displays just how immense the amount of data in the Steam Reviews dataset is. Figure 2 below visually displays the vast difference in review quantity between the most popular games and the least popular games. For filtering, we will use a simple comparison expression and just delete games with less than 10000 reviews out of the dataset that we store for processing.

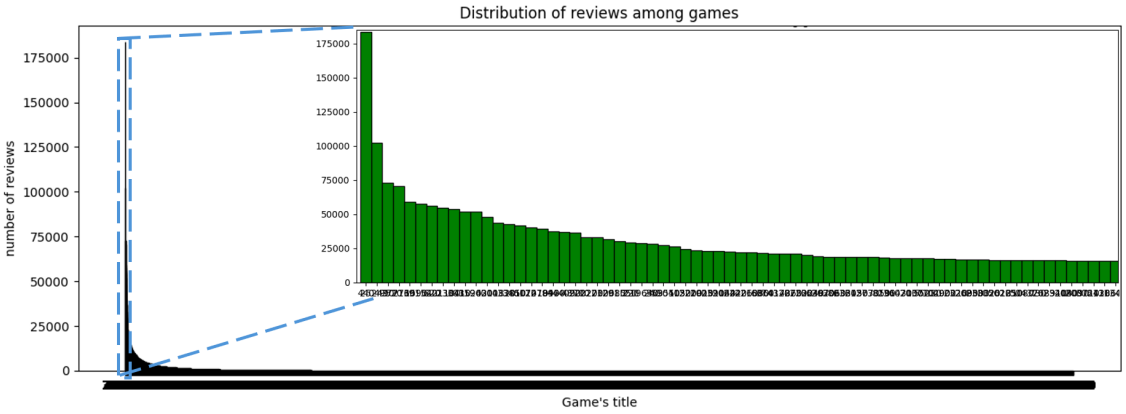

Figure 2: Distribution of Steam Reviews from Most Reviews to Least Reviews

From here we focused on the 10 most popular games determined by the most number of reviews. We then changed their game ID to the exact title of the game.

The following details the steps we took in generating a summary ideas for each of those 10 games.

2.2 Cleaning Up the Data

The structure we had is that a list called "response" with its items are reviews from users. We began by removing any reviews that consisted of less than 55 words. These reviews did not contain enough substance to contribute meaningful insight toward our final summarization, so we decided that the best course of action was to remove them. Then, for each user's review, we separated them into sentences by period. The list now contains review sentences as its items.

From there we made sure to expand any contractions before we applied any word pairings. This database is written in Python code from a Wikipedia list of contracted words [1]. If the word has more than one expanded form, we just simply select the first option.

2.3 Part of Speech Tagging + Sentiment Analysis

From this point, we processed each sentence individually. In the loop which iterated through all sentences in the "response" list, we tokenized the sentences into words and assigned tags indicating their word form. There are 4 forms that we are interested in noun, verb, adjective, and adverb. Alongside this tokenized process, we got the sentence through the "get sentiment" function made from the package "nltk.sentiment.vader". Tokenized, tag assignment, and sentiment.vader are functions from Natural Language Toolkit (nltk). VADER (Valence Aware Dictionary and Sentiment Reasoner) is a built-in pre-trained sentiment analyzer of NLTK. This VADER analyzer has a method to output

4 scores ranging from 0-1, compound, positive, negative, and neutral, of the sentence. In our "get sentiment" function, we use the analyzer on the sentence and get its compound score, which is the average of the other 3; if the compound score is bigger than 0 then we assign a score of 1 for the sentence, and vice versa we assign a score of 0 to the sentence.

2.4 Pair Words

Then, we iterated through the tagged tokens in each sentence by pair. The idea is that Word[0] is paired with Word[1], Word[1] is paired with word[2], and so on. However if a word doesn't belong in the 4 categories, we skip the pair that has them. There are 6 pair lists, which are combinations of the 4 categories: noun-adj, noun-verb, noun-adv, verb-adj, verb-adverb, and adj-adv. To summarize, the tokens were grouped into tuples by the structure (sentiment score of the sentence, word 1, word 2). This tuple would then have their word order sorted alphabetically to ensure there were no duplications, like (1, "game", "fun") and (1, "fun", "game") will be normalized to (1, "fun", "game") and sorted to noun-adj list. Finally, coming out of the processing function is a dictionary with 6 lists of word pairs and their sentiment score.

2.5 Clustering Kmean Clusters by Cosine Similarity and Euclidean distance

Inspired by the work done by Thakur, we used the pre-train GloVe model "Glove.6B.100d" for all places that need a word vector model [7]. It was big enough for informal texts and not too big that it would take forever to load it. The idea for clustering is to get all the word pairs on the mathematic plane using the Glove pre-trained model. By calculating the position of those word pairs, we can know how most reviewers judge the game. More words at the center of the cluster mean that those are the main ideas of users. To calculate the cluster center more easily, we used the K-mean calculation when we clustered the word pairs. At this point, we had 2 clusters for each word pair list.

We used K-mean to calculate the location for the center of the cluster, which was then used to calculate the distance of the word pairs from the center. There are two methods of calculating the distance in this project, Cosine Similarity calculates the angle form between the line from center to two word pairs; Euclidean distance calculates the linear distance difference between 2 vectors of the word pairs. We implemented a parallel thread in this part to calculate these two simultaneously and reduce computation time. The calculated distances are stored in a list sorted in ascending order. For this result, we picked 8 pairs of words from each cluster that has the smallest angle or distance differences from the center. There is a unique filter here that ensures no word pair is repetitively picked, so if (1, "game", "fun") is already selected, another (1, "game", "fun") near that place cannot be picked.

3 Results

The main idea word pairs for the game "Rust" computed by 2 methods would look like the output that is located on page 5-8.

Pairs Computed Using Euclidean Distance for Rust

Category: Noun-Adjective

Cluster 1:
(1, 'get', 'moreover')
(1, 'instance', 'same')
(0, 'folks', 'only')
(1, 'addition', 'good')
(0, 'anymore', 'same')
(1, 'anymore', 'same')
(0, 'headshots', 'most')
(1, 'full', 'ones')
Cluster 0:
(1, 'first', 'game')
(0, 'first', 'game')
(1, 'end', 'game')
(0, 'end', 'game')
(1, 'finished', 'game')
(0, 'finished', 'game')
(1, 'fantastic', 'game')
(0, 'fantastic', 'game')

Category: Noun-Verb

Cluster 1:
(1, 'eachother', 'is')
(1, 'admins', 'put')
(1, 'headshots', 'is')
(0, 'admins', 'having')
(0, 'abit', 'be')
(1, 'headshots', 'make')
(0, 'admins', 'being')
(0, 'admins', 'coming')
Cluster 0:
(1, 'be', 'game')
(0, 'be', 'game')
(0, 'game', 'say')
(1, 'game', 'say')
(1, 'game', 'have')
(0, 'game', 'have')
(1, 'game', 'want')
(1, 'come', 'game')

Category: Noun-Adverb

Cluster 1:
(1, 'again', 'kind')
(1, 'everyone', 'well')
(0, 'example', 'just')
(1, '[', 'even')
(0, 'instead', 'nothing')
(1, 'even', 'turn')
(0, 'everyone', 'however')
(0, 'kind', 'later')
Cluster 0:
(1, 'even', 'game')
(0, 'even', 'game')
(0, 'either', 'game')
(0, 'else', 'game')
(1, 'definitely', 'game')
(0, 'definitely', 'game')
(0, 'enough', 'game')
(1, 'enough', 'game')

Category: Verb-Adjective

Cluster 1:
(0, 'get', 'is')
(1, 'get', 'is')
(1, 'be', 'little')
(0, 'be', 'little')
(0, 'be', 'much')
(1, 'be', 'much')
(1, 'be', 'good')
(0, 'be', 'good')
Cluster 0:
(1, 'make', 'sigle')
(1, 'good', 'modded')
(1, 'good', 'recieve')
(1, '0-25', 'is')
(0, 'hmmmmm', 'is')
(1, 'abit', 'get')
(1, 'know', 'wich')
(0, '+60', 'get')

Category: verb-adverb

Cluster 0:
(1, 'even', 'supposed')
(0, "'d", 'rather')
(0, 'come', 'simply')
(1, 'come', 'obviously')
(1, 'forget', 'well')
(1, 'coming', 'simply')
(1, 'even', 'realize')
(1, 'knowing', 'well')
Cluster 1:
(1, 'is', 'not')
(0, 'is', 'not')

(0, 'has', 'not')
(1, 'has', 'not')
(0, 'been', 'not')
(1, 'been', 'not')
(1, 'as', 'be')
(0, 'as', 'be')

Category: adjective-adverb

Cluster 0:
(0, 'glitchy', 'so')
(1, 'glitchy', 'so')
(1, 'extremly', 'good')
(0, 'amazingly', 'good')
(1, 'amazingly', 'good')
(1, 'admittedly', 'good')
(1, 'fun', 'yet')
(1, 'good', 'sooooo')
Cluster 1:
(0, 'much', 'so')
(1, 'much', 'so')
(0, 'more', 'once')
(1, 'more', 'once')
(1, 'as', 'much')
(0, 'as', 'much')
(0, 'instead', 'much')
(1, 'however', 'much')

Pairs Computed Using Cosine Similarity for Rust

Category: noun-adjective

Cluster 1:
(1, 'good', 'one')
(0, 'good', 'one')
(1, 'kind', 'only')
(0, 'kind', 'other')
(1, 'kind', 'other')
(1, 'little', 'well')
(1, 'kind', 'same')
(0, 'kind', 'same')
Cluster 0:
(1, 'first', 'game')
(0, 'first', 'game')
(1, 'end', 'game')
(0, 'end', 'game')
(1, 'finished', 'game')
(0, 'finished', 'game')
(1, 'fantastic', 'game')
(0, 'fantastic', 'game')

Category: noun-verb

Cluster 1:
(0, 'going', 'use')
(0, 'find', 'one')
(1, 'find', 'one')
(0, 'find', 'time')
(1, 'find', 'time')
(0, 'go', 'people')
(1, 'go', 'people')
(1, 'going', 'people')
Cluster 0:
(1, 'be', 'game')
(0, 'be', 'game')
(1, 'game', 'have')
(0, 'game', 'have')
(1, 'be', 'play')
(0, 'game', 'say')
(1, 'game', 'say')
(0, 'do', 'game')

Category: noun-adverb

Cluster 1:
(1, 'even', 'way')
(0, 'even', 'way')
(1, 'even', 'time')
(0, 'even', 'time')
(1, 'get', 'well')
(0, 'even', 'get')
(1, 'again', 'kind')
(0, 'get', 'so')
Cluster 0:
(0, 'either', 'game')
(1, 'even', 'game')
(0, 'even', 'game')
(1, 'definitely', 'game')
(0, 'definitely', 'game')
(0, 'else', 'game')
(0, 'essentially', 'game')
(1, 'especially', 'game')

Category: verb-adjective

Cluster 1:
(1, 'be', 'good')
(0, 'be', 'good')
(1, 'be', 'little')
(0, 'be', 'little')
(0, 'be', 'much')
(1, 'be', 'much')
(0, 'get', 'is')
(1, 'get', 'is')
Cluster 0:
(0, "'re", 'only')
(1, 'know', 'little')
(1, 'know', 'simple')
(1, 'do', 'little')
(1, 'crazy', 'get')
(0, "'re", 'bad')
(1, 'feel', 'good')
(0, 'feel', 'good')

Category: verb-adverb

Cluster 0:
(1, 'even', 'going')
(0, 'even', 'going')
(1, 'do', 'just')
(0, 'do', 'just')
(1, 'come', 'so')
(0, 'come', 'so')
(1, 'even', 'go')
(0, 'even', 'go')
Cluster 1:
(1, 'is', 'not')
(0, 'is', 'not')
(0, 'has', 'not')
(1, 'has', 'not')
(0, 'been', 'not')
(1, 'been', 'not')
(1, 'as', 'be')
(0, 'as', 'be')

Category: adjective-adverb

Cluster 0:
(1, 'fun', 'very')
(0, 'fun', 'very')
(1, 'fun', 'good')
(1, 'fun', 'yet')
(1, 'extremely', 'fun')
(0, 'extremely', 'fun')
(0, 'good', 'really')
(1, 'good', 'really')
Cluster 1:
(0, 'much', 'so')
(1, 'much', 'so')
(1, 'much', 'not')
(0, 'much', 'not')
(1, 'more', 'not')
(0, 'more', 'not')
(1, 'most', 'so')
(0, 'most', 'so')

Result Evaluation: Since there is no gold standard summary (a pre-existing summary that gives an accurate depiction of the games available on Steam) to compare generated summaries with, it is known from past work in this area that the standard protocol for evaluation has been experienced user study evaluation [2]. However, given the time constraints of our project, the conduction of a proper, full-length user-study was not possible, so we proceeded with manual evaluation of our work. One of our authors, Bao, has experienced knowledge in video games and was able to provide background insight on the video games and the general user consensus when reviewing our results for accuracy. In addition, another component of manual evaluation is the readability of the results, that is how well the results can be comprehended by the person reading them.

When conducting our manual evaluation, we drew several conclusions. First, we had originally tried to implement the most frequent words from each category (verb, adj, adverb, noun) into a predefined sentence template and that was not a very successful option given the readability of the outputted sentences was quite unclear and not helpful. Thus, we sought outside expert advice from Dr. Bible for ways to improve upon the generated results. Taking his insight, we revised some of our work to include GloVe, Clustering and Pair Words. Now the focus was not on the generation of sentences, but on generating pairs of words. This was because we felt that assessing the co-occurrence of two words would provide solid insight into the underlying meaning and sentiment of the user review without

having to read the entirety of the review. This keyword extraction was a way for us to find recurrent themes in the sentiments of the users in a more efficient way. In evaluation of the outputted word pairs from the game Rust, we made several observations on our results through our manual evaluation. First, we used both Euclidean Distance and Cosine Similarity to produce pair words. In evaluating whether one produced a better result than the other, we concluded that they was negligible difference between the outputs. We conclude this to be the case because of the amount of words we picked out compared to what exists in the database vector plane. Then, when reading the pairs of words per each category, we observed that the relevance and usefulness varies by category depending on what the tagged pair consists of i.e. noun-adj etc.. For example, we found that the category of adj-adverb produced a very useful result. This is because we identified word pairs such as extremely with fun, or very with fun which provides a beneficial summary of the sentiments of the users about that given game. In addition, there was also identified usefulness in the noun-adj category. We pulled pair-words such as "fantastic game" which also provides beneficial feedback on the users' sentiments. On the reverse end, we also identified category pairs that do not add as much benefit in summarizing the review. For example, the verb-adverb and noun-verb categories produced pairs of words that do not provide much insight into telling us about the game or the users' sentiments. Thus, while we successfully identified multiple pair-word categories, we conclude that there exists a varying degree of benefit between each categories with categories consisting of an adjective providing more beneficial insight and categories consisting of a verb providing the least amount of beneficial insight.

In addition to the pairs of words, we also included a sentiment score that notes whether the words come from a positive or negative sentence. 1 meaning the overall sentiment of the origin sentence was positive and 0 meaning the overall sentiment of the origin sentence was negative. While the words themselves especially adjectives (describing words) help to allude to the sentiment, there exists a few reasons why this score is necessary and useful. First, in the adj-noun category, we observe an output such as (1, "good", "game) and (0, "good", "game"). Here we have the same pair words, yet one originates from a positive sentence and the other negative. We can conclude that the second pairing either comes from a sarcastic sentence (so it is suggesting the opposite of what the words mean) or that the sentence includes a contradictory conjunction such as "however" that introduces a small positive thought after a previous negative sentiment. So while the number itself does not reveal the exact scenario of the ones listed above, it serves more so as a warning label for the user reading the word pairs.

4 Discussion

While product review summarization is not a novel concept, it remains an ever-evolving technique as more and more companies are gathering review data from their users. Our project sought to use product summarization techniques in a way that both benefits the users and the company itself contributing to greater user satisfaction and improved Steam performance.

Discussion of our technique + future work:

Admittedly, there are a few limitations to our analysis including time constraints, no user study and a lack of gold-standard summaries to compare our generated summaries with. As much as we would like to take into account all the possible sentences that could result from the contractions. For example, if a sentence contained the word "I'd", we expanded the sentence to two versions. One had "I had" while the other had "I would" in the sentence. A similar fashion to the techniques outlined by Thakur's work in text summarization, converting the sentences to vectors using GloVe [7] and using Cosine similarity to compute their similarity to each other. The function would then pick the one with the highest score to be the version of expansion for the sentence with contracted word(s). However, this method here is too computing demanding as a user can write paragraphs of reviews and most of the sentences in there have some form of contracted word. We deemed performing Cosine similarity

on such a large scale with little return was not worth the effort.

When picking the number of clusters for each word pair category, we tried 3 clusters initially but received warning that the number of unique vectors is smaller than the number of clusters. We suspect that the pre-trained model may be the reason for this. However, we haven't figured out how to train the model with our type of data due to time constraint of a semester project. Doing the clustering process with our trained model would definitely yield more relevant result than the general pre-trained model.

The part that took the longest to finish was the pre-processing sentence to cluster. The process ranges from converting list of users' reviews to 6 lists of word pairs. This part takes around a quarter of the processing time. The Clustering process takes up about more than half of the processing time. We only had the calculation of cosine similarity and euclidean distance on parallel thread. In the future, finding a way to make the whole process runs on parallel thread would significantly reduce the waiting time. In addition, future work would involve creating some validation function to automatically evaluate the game overall sentiment score (average of all sentiment scores in the game) and change the amount of pairs to select. For example a game has a score of 0.8 (0-1), it will then have 6 or 7 pairs of word to be positive and the rest remain negative.

We also spent a great deal of time on pre-processing of the data. The large scale of game reviews meant challenges exist for highlighting the content a user will find compelling. Within the dataset, there is a wide variation in the amount of reviews generated for each product. This disparity in review quantity made it necessary to detect the products with the greatest amount of reviews as this corresponds with the popularity of the product. This way we could ensure we were efficiently spending time addressing the biggest products, while spending less time on products that have produced little interest in the consumer reviews. Besides, the games with low reviews can be handle by group of people. Our scope of this project was on games with massive amount of reviews, which would be easier and more convenient for machines to handle. Thus, it was necessary to pre-process the data accordingly.

Furthermore, evaluation of this work in the future would require the development of a dataset of gold standard summaries. The analysis would involve randomly removing a sentence from the original review and generating a way to measure if we still retain similar word pairing results to the gold standard summary and our original word pairings. The goal would be that small random removal of review sentences would not significantly alter our word pairing results. This would allow the model to be able to have some self-evaluation to help developers who want to improve the summaries generated.

Ultimately, in conclusion, we believe that our overall process for Steam review summarization was reasonably effective. While there is always room for improvement (which we identified above in results and discussion), we believe that some of the outputted word-pairing categories provides beneficial insight into the general consensus of sentiments expressed in the user review data. Within the popular product reviews, we sought to formulate a condensed, summarized version with word pairs that encompasses all the feedback and sentiments expressed in the reviews. Providing a user with a summarized list of word pairings over a massive list of individual reviews will serve to increase user satisfaction as they do not have to search hard for the answers they are searching to discover. In addition, a summarization of users' sentiments regarding certain products would serve to benefit Steam by helping craft a more pointed course of action in product performance and promotional strategy since it would be tailored to the general consensus of users found in the reviews. This tailoring could help improve sales and registrations. This would also lead to easier identification of poor product response and poor service performance. The ability to resolve and/or address such issues in a timely manner would result in fewer user departures to a competing platform. Efficient review summarization of Steam reviews would prove to be a beneficial tactic for improving user satisfaction, platform performance, and product promotion, all of which would work to boost overall revenue.

References

[1] Nov. 2024. URL: https://en.wikipedia.org/wiki/Wikipedia:List_of_English_contractions.

[2] Aris Kosmopoulos et al. "Summarizing Game Reviews: First Contact." In: *SETN Workshops*. 2020, pp. 22–31.

[3] Nicholas W Landry et al. "Filtering higher-order datasets". In: *Journal of Physics: Complexity* 5.1 (2024), p. 015006.

[4] Duy Khang Ly et al. "Product review summarization from a deeper perspective". In: *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*. 2011, pp. 311–314.

[5] George D Panagiotopoulos, George Giannakopoulos, and Antonios Liapis. "Automatic Summarization of Video Game Reviews". In: (2019).

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[7] Isha Thakur. *Textsummarizer using TextRank and glove embeddings*. Aug. 2020. URL: https://itsmeishathakur.medium.com/textsummarizer-using-textrank-and-glove-embeddings-59a229a98966.

[8] Mengting Wan and Julian McAuley. "Item recommendation on monotonic behavior chains". In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 86–94.

[9] Kevin Yauris and Masayu Leylia Khodra. "Aspect-based summarization for game review using double propagation". In: *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*. IEEE. 2017, pp. 1–6.

[10] Wen-Jie Ye and Anthony JT Lee. "Mining sentiment tendencies and summaries from consumer reviews". In: *Information Systems and e-Business Management* 19.1 (2021), pp. 107–135.

[11] Jing Zhao et al. "An Improved K-Means Algorithm Based on Contour Similarity". In: *Optimization Algorithms in Data Science: Methods and Theory* (2024).