Steam Scraper Project Proposal

Rob Bray, Matthew Dixson, Tom Hills, Tan Nguyen

***Abstract*— Video games are becoming increasingly popular. We will explore how much a video game's genre affects different aspects of the game, such as rating, price, discounts, extra downloadable content, and computer requirements.**

I. Objectives

Steam is a popular online store for electronic entertainment which primarily consists of video games. Because of Steam's popularity and importance as a large distributor of electronic media, we wish to gain information about the applications sold on Steam. Primarily, we are interested in prices of apps, genres and tags associated with apps, descriptions, downloadable content (DLC) prices, user ratings, and apps' system requirements. We will look at the DLC and user ratings through the lens of genre in order to determine if there are any notable differences persisting across genres.

II. Motivation

Steam is one of the largest retailers for video games on PCs. Steam also sells games for a discounted price quite often. Knowing when games go on sale, how much their price is discounted when they go on sale, and knowing a game's user ratings are important to video game enthusiasts, because these things influence if they will purchase a game or not. It would also be beneficial to buyers to know this data and more by video game genre. With this information, a buyer can look in the genre with high average rating and large discounts for highly rated video games that are on sale.

III. Data Collection

Our project involved gathering data from Steam's online storefront. In order to accomplish this goal, we first made use of Steam's official API. This API allowed us to easily get a list of 127,239 applications with ID-title mappings.

This mapping proved to be invaluable to us. Firstly, having a list of app titles allowed us to filter out certain applications. This was particularly useful as a few games were inappropriate and had adult content. We were able to remove inappropriate apps from consideration with a blacklist of words. This filter was not entirely effective as some games with adult content were left in the list, but it was successful at removing the most inappropriate apps.

However, we were not able to get any other useful information using Steam's API. Thus, we moved on to other methods. In particular, we used the Python library BeautifulSoup4 to implement web scrapers to get data directly from the Steam storefront. This is where the ID portion of the aforementioned ID-title mapping became useful. All of Steam's store pages have the following syntax: store.steampowered.com/app/id. There is only one variable in this URL that we had to pay attention to: id. Putting an app's ID in this field allows one to navigate to the corresponding web page. For example, using 489830 will bring one to Steam's store page for *The Elder Scrolls V: Skyrim Special Edition*. By combining our list of IDs with BeautifulSoup4, we were able to automate web scraping of Steam and collect various types of data. We ran our web scrapers around the beginning of November, 2021.

Due to possible execution interruptions from timeouts during a page request, scraping for genres vs. counts and price ranges vs. counts used a "stop/resume" functionality which allowed the program to track the last Steam app processed and resume scraping on the next one in the list. This was essential for sequentially processing the list as the total execution time for sequentially going through each app was around 22 hours.

We used Python's multiprocessing library in order to reduce the amount of time spent on scraping data. This involved writing a web scraping function and using multiprocessing's Pool object to spawn multiple processes. Each process then scraped specific data, such as DLC titles and prices, from a single Steam page. This data was then saved into comma separated values (CSV) files. This method greatly increased the speed of data collection. It only took a few hours to get a significant amount of results.

We experienced very few problems while scraping with multiprocessing. Steam stopped us from connecting twice while using this method. This ban was only for a short period of time and it was only after scraping for a long period or by spawning too many processes.

IV. Datasets

A. Genre/Price Range & Counts

The dataset for this analysis was saved in a CSV file and contained 111,112 entries. Each line contained the line number from the original request list, appID, app title, a single tag, app genre, and app price. With this structure, all information was repeated per line except for the tags, which were listed one per line. This organization was an attempt to mimic SQL cross product arrangement to make it easier to use Excel to filter items and create graphs. For Genres, all 111,112 entries were used in the analysis while only 80,838 entries were used in the price range analysis.

B. DLC

The DLC dataset was saved in a CSV file. Only apps with DLC were in this data set. Each game was on its own line. Each line then had an app's name, ID and all DLC titles immediately followed by their price and the percentage

discount if the DLC was on sale. In total, there were 4142 entries in this data set.

C. User Reviews

The user review data was also saved in a CSV file. This dataset consists of 80,567 entries from our ID-Mapping list. We were not able to get reviews for all 127,239 entries in our ID-Mapping list because not all apps have user reviews and we were temporarily banned from connecting to Steam while collecting data for this dataset. The CSV file for the user reviews dataset had one entry per line. Each entry consisted of the app's title, ID, and the average rating. Ratings are described with text, such as "positive" or "negative", by Steam.

D. Genre & Descriptions

There were 113,902 entries found for genre and description. Since this analysis was only concerned with looking at descriptions against genres, the descriptions for games without a genre specified, or the inverse for genres, were removed from the set. However, most games were kept in the set. Descriptions were difficult to find, since there were several instances where the description text appears in the HTML that are not easy to get. A lot of trials were spent trying to extract from this place in the HTML. However, the class element that held the text, and was easy to extract, was found after doing more searching through the list of instances.

E. Minimum and Recommended System Requirements

This dataset was saved in two text files named test_minimum.txt and test_recommended.txt . Each game is separated by a line of 36 c̆haracters. The first line is the name of the video game and its ID. Next lines are followed by OS, processor, memory, graphics, directX, and storage. There are some games that did not show some variables because they did not show up on the website. A total 77401 entries are in this file.

V. Data Processing

A. Genre/Price Range & Counts

Due to the CSV data being organized in a SQL cross-product style, Excel was used to both analyze and graph the data. In order to properly analyze the genre and prices, the original data table was copied, the tags column removed, and the repeating data from the other columns condensed down to one appID per line. Originally, there were ideas on how to incorporate the tag data in these, or other analysis, but after reviewing the tags, there didn't seem to be a way to analyze them in any meaningful way.

The Genre vs. Count analysis was accomplished by searching through the "genre" column and picking out all unique genre labels. Once completed, each unique genre label was used in a search and count algorithm to count the number of times an appID was labeled as such. A table was composed from the unique genres and associated counts and a bar graph was created to visualize the results.

The Price Range vs. Count analysis was accomplished by first filtering the "price" column to eliminate data that was not a price. Once completed, a table of price ranges (minimum and maximum interval prices) was created which was then used to search and count the number of appIDs whose price fell between these boundaries. A new column of counts was added to the existing table which was then used to create a bar graph to visualize the results.

B. DLC

The DLC dataset was processed in Python using numpy, scipy, and matplotlib. Firstly, the DLC dataset was parsed in order to extract the price of the DLC. All prices had a preceding "$", so this was used to locate prices. Some DLCs were free and had the word "Free" or "free" in place of a numerical price. A price of free was interpreted as $0. After extracting all DLC prices, scipy was used to obtain various statistics about DLC price. Such statistics include, mean, median, mode, and standard deviation. DLCs in certain price ranges were also counted. Some DLCs were on sale at the time of scraping, so we used the discounted price.

All DLCs that were on sale also included a percentage discount. This allowed us to analyze and process DLC discounts as well as price. We performed a similar operation for discounts as we did for DLC price. We determined the same statistics as well. We also looked at DLC information that belonged to a certain genre in addition to looking at DLC of all games. In particular, we examined the DLC of the five genres with the most apps. The same analysis as described above was carried out for each five genres. This process is described in more detail in the data integration section of this paper.

C. User Reviews

Once a user buys an app on Steam, they are able to leave a review for that app and rate it. Steam determines what the average rating for an app is on a per-app basis. These averages are texted based. For example, if almost all users leave a positive review, Steam will say that the reviews are "Overwhelmingly Positive". The text descriptors for reviews from most positive to least positive are Overwhelmingly Positive , Very Positive, Positive / Mostly Positive (Difference based on number of reviews. Positive means fewer reviews), Mixed, Negative / Mostly Negative (Same distinction as positive), Very Negative, and Overwhelmingly Negative. In order to perform statistical analysis for user reviews, this rating system needed to be converted into a numerical system. For our purposes, we used a 0-6 scale. Overwhelmingly Negative as assigned 0 and Overwhelming Positive was assigned a 6. Thus, the higher the rating number is, the more positive an app's reviews are.

Some apps had very few reviews and Steam did not give these a descriptor. Instead, Steam just told how many reviews there were. We ignored these apps.

We performed an analysis similar to our DLC analysis for user reviews. Statistical data such as the mean, median, and mode rating was evaluated. We also totaled how many

apps had each rating. We then looked at user reviews of the 5 most numerous genres. The genre-user review analysis is described in more detail in the data integration section of this paper.

D. Tagging Descriptions

In order to tag words for type, the Natural Language Toolkit (NLTK) was used. Inside of NLTK, the Part of Speech (POS) tagger was used to train and classify these words. This part of the analysis was concerned specifically with the proportion of nouns, verbs, and adjectives to the total words in each genre.

E. Counting Words of Interest

For the word counts of each word-of-interest, the data first needed to be cleaned up in order to avoid counting uninteresting words, like 'a,' 'the,' 'you,' etc. This was done using the NLTK corpus stopwords() function. From there, the program simply used a python Counter() object in order to determine the most-used words. An interesting observation was that many genres had the same words in their top-ten most-used words, like 'play,' 'world,' and 'new.' The program then calculated the proportion of these instances of words in the totality of words collected for each genre.

F. Minimum and Recommended System Requirements

There is a limitation on this task. The data could not be statistical because the authors entered very different information for different games. For example, on games "Fernbus simulator - France, 654120" and "Realms of Arkania, 654140", the OS is Windows 7 slash 8 slash 10 compared to Windows 7 comma 8 comma 10. This problem can be solved with filter function, but no, it still has problems with other variables like processor and graphics.

VI. Data Integration

We integrated our DLC, user review, and description datasets with our genre dataset. For the DLC and user review datasets, we used genre as a filter. We created a script that checked if an app was in a genre by using an ID-genre mapping. If the app was in the specified genre, then it was included in the analysis. Otherwise it would be left out. We examined DLC and user reviews in relation to five of the most numerous genres: action, adventure, casual, indie, and simulation. There was a problem when integrating the user review dataset with the genre dataset because not every genre had one of every rating. For example, some did not have overwhelmingly negative ratings so this had to be taken into consideration while doing analyses. We examined descriptions as they relate to genres by looking solely at games that had both a genre labeled and a description written. If the game did not have both of these, then it was not considered for analysis.

VII. Models Used

A. Genre & Description

The POS tagger, pos_tag(), from NLTK is an averaged perceptron, where the tags are the classes. During training, the features are extracted from the words, the weights are added to these features, and the training determines if the prediction was correct. If it was incorrect, the model punishes this set of weights by subtracting from it, and adding to the set of weights that had the correct tag. The model returns the averaged weights in order to make the algorithm more resistant to variance in training sets [1].

VIII. Data Validation

All of our data came directly from Steam. As such, we have reason to believe that our collected datasets are representative of Steam games. Despite this some of the entries in our datasets were not useful. For example, some entries would have empty strings for some fields making analysing them difficult. We dealt with these entries by simply dropping them from consideration. Most of the time, the number dropped was not a significant number. The dataset that we had to drop the most results from was the user review dataset. We considered 43,652 entries in our analyses involving this dataset and dropped 36,896. These entries were largely dropped to have very few reviews and were not simply useful for our analyses. This left a set of entries that was still quite large. With these drops in mind, we would still argue that our results are valid for Steam games.

IX. Results

A. Genre & Count

This analysis focused on counting how many apps were labeled with a particular genre. The results of this effort can be seen in figure 1 below:

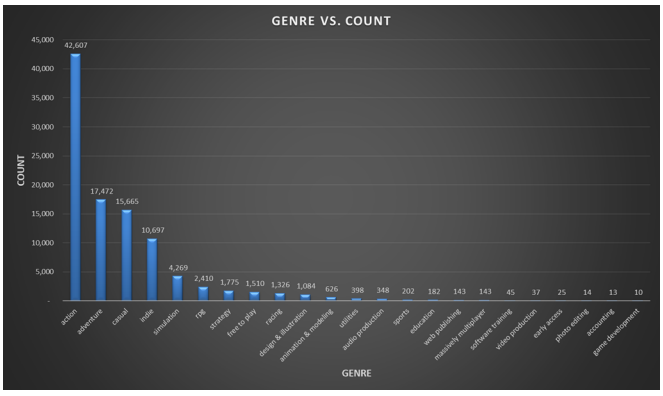

Fig. 1. Genre vs. Application Count

The x-axis of figure 1 lists every genre found while scraping and the y-axis shows the number of applications labeled with the associated genre. One interesting observation is that Steam doesn't just sell games, but other software as well. This can be seen with genres like "design illustration", "audio production", "web publishing", and others. Another

observation is that the genre for some familiar games doesn't seem to match with what is expected. Many classical RPG games were labeled as "adventure", for instance. Additionally, there were genres which were not easy to define. Genres like "casual", "early access", "indie" and others didn't seem to indicate what kind of game or app they were associated with. With all this taken together, the genre does not seem to be as useful as originally thought when it comes to game classification.

B. Price Range & Count

This analysis focused on counting how many apps had prices which fell into prescribed price ranges. The results of this effort can be seen in figure 2 below:

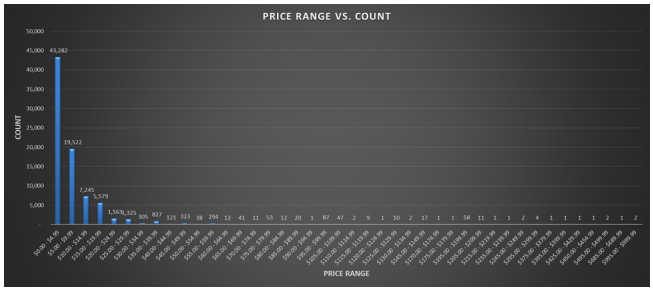

Fig. 2. Price Range vs. Application Count

The x-axis of Figure PRC shows price ranges in spans of 5 dollar increments and the y-axis shows the number of applications with prices falling in the associated range. Since not all Steam store web pages shared the same HTML design when displaying price (though luckily, most did), prices for only 80,838 applications out of roughly 130,000 were collected for this study. In figure 2, you'll notice that some price ranges are not shown which indicates that no applications were found to be in that price range. From analyzing the data, it was calculated that roughly 77% of the applications had prices of $9.99 or less while 99.5% sold for $59.99 or less. The minimum selling price was found to be $0.49, the maximum price was $999.00, the average price was $8.85, and the median price was $4.99. With the average and median prices being roughly comparable, it is likely that distribution of selling prices is appropriately uniform, though it's definitely skewed somewhat towards the lower prices as figure 2 clearly shows.

C. DLC

Prices. The results of our statistical analysis of DLC prices are shown in figure 3. A combination of all genres had a mean price of $6.09, a median of $3.99, and a mode of $4.99. These figures are useful as they determine what the average DLC will cost and what the most common DLC price is. This set also has a standard deviation of around 13, meaning that most DLCs' prices lie within $13 dollars of $6.09. The minimum price of the dataset was $0 and the highest price was $450.

	All Genres	Action	Adventure	Casual	Indie	Simulation
Mean ($)	6.09	6.41	4.48	4.38	4.75	9.17
Median ($)	3.99	3.99	2.99	1.99	2.99	6.99
Mode ($)	4.99	4.99	4.99	0.99	4.99	9.99
Mode Count	1052	454	257	187	102	36
Std Deviation	12.996	12.568	7.84	9.128	7.004	7.989
Min ($)	0	0	0	0	0	0
Max ($)	450	199.99	199.99	199.99	119.99	39.99
Total	6918	2951	1557	950	624	172
Dropped	3	0	3	0	0	0

Fig. 3. Statistical Information for DLC Prices by Genre

However, these statistics tend to change from genre to genre. In particular, action and simulation DLC is more expensive on average when compared to the all genres average. The simulation average is quite a bit more being $9.17. The remaining three genres, adventure, casual, and indie, all have averages lower than the all genres average. Simulation also has higher median and mode prices as well. However, the highest simulation DLC price ($39.99) is significantly lower than the other genres' highest prices (all over $100). The standard deviation for action DLC is close to the all genres standard deviation while the other genres have smaller standard deviations.

Figure 4 shows how many DLCs fall into certain price ranges capped at $50. Almost 5000 DLCs fall in the $0-5 price range. Over a 1000 DLCs fall in the $6-10 price range. After this, the number of DLCs in each price range falls significantly. Only a few hundred are in the $10-20 range and even less in the ranges after this. There were DLCs with prices over $50, but there were so few that they were left out.

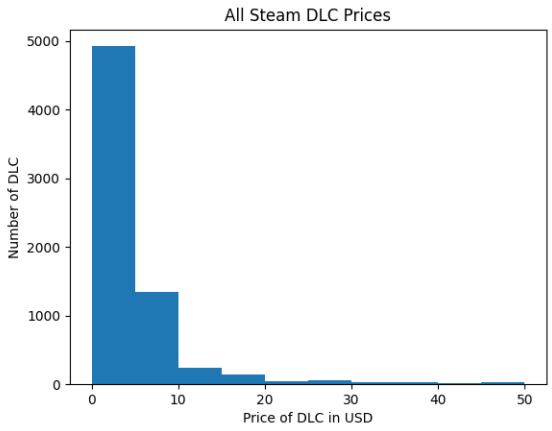

Fig. 4. Steam DLC Prices

Discounts. Some DLCs were sold on sale. We performed a similar statistical analysis with these specific DLCs as with DLC price. In particular, we looked at the percentage discount for each of these DLCS. Figure 5 shows the results of this analysis.

	All Genres	Action	Adventure	Casual	Indie	Simulation
Mean (%)	49.23	50.73	47.4	48.03	47.07	27
Median (%)	50	51	51	51	50	25
Mode (%)	51	51	51	51	50	5
Mode Count	104	18	51	29	18	1
Std Deviation	18.334	23.316	12.927	14.251	15	18.833
Min (%)	5	10	10	10	10	5
Max (%)	100	100	80	90	90	51
Total	383	133	111	67	45	3

Fig. 5. Statistical Information for DLC Discounts by Genre

DLCs in all genres had a mean, median, and mode of around 50%. This set also had a standard deviation of 18.334. This means that most discounts were 18 percentage points away from 50%. The smallest discount was 5% while the largest was 100%. The action, adventure, casual, and indie genres had very similar results to all genres. The biggest difference was that the standard deviation for action was significantly larger while the standard deviation for the other three was less. Simulation was the most different. It had roughly the same standard deviation, but it had a mean discount of 27%, a median of 25%, and a mode of 5%. It is important to note that the sample size for the simulation set is only 3. This indicates that simulation DLCs go on sale more rarely than other genres.

Figure 6 shows the number of DLCs of all genres that fall into a discount range. Most discounted DLCs fall into the 40-60% off range. Almost equal numbers fall into the 20-40% and 60-80% ranges and the 0-20% and 80-100% ranges.

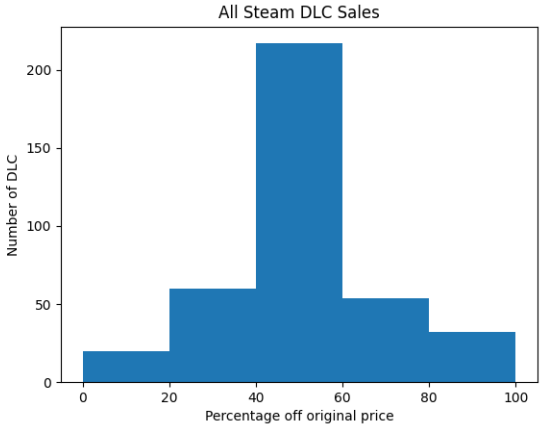

Fig. 6. Steam DLC Discounts

D. User Reviews

As mentioned above in the data processing section, Steam generates a text rating descriptor from user reviews. We used these text rating descriptors to form a numerical rating system and used this to perform statistical analyses of ratings based on genre. Figure 7 shows the results of these analyses.

Apps from all genres had a mean rating of 3.99, a median rating of 4, and a mode of 4. This shows that several users tend to give games positive reviews. The standard deviation of this set is rather small- only .88777. This means that most ratings are mixed, positive, or very positive, which results from users giving positive reviews. The specific genres follow these results almost exactly. There is very little variation in the mean and standard deviation for each genre. None of genres' means differs from the all genre mean by more than .5. This further reinforces the claim that most users give positive reviews.

	All Genres	Action	Adventure	Casual	Indie	Simulation
Mean	3.99	3.95	4.1	3.96	4.02	3.66
Median	4	4	4	4	4	4
Mode	4	4	4	4	4	4
Mode Count	18373	7039	3361	2609	1541	778
Std Deviation	0.88777	0.9	0.88564	0.8577	0.92284	0.89637
Min	0	0	1	1	0	0
Max	6	6	6	6	6	6
Total	43652	17283	8239	5671	3986	2022
Dropped	36896	13457	4960	5918	2888	1363

Fig. 7. Statistical Information for App Ratings by Genre

Figure 8 below shows the number of apps that have been given a certain rating. Most apps received a rating of positive or higher. Several games also received mixed ratings. Very few games received a negative rating, yet alone a very negative or overwhelmingly negative rating. In fact, only 8 games out of around 43,000 received an overwhelmingly negative rating.

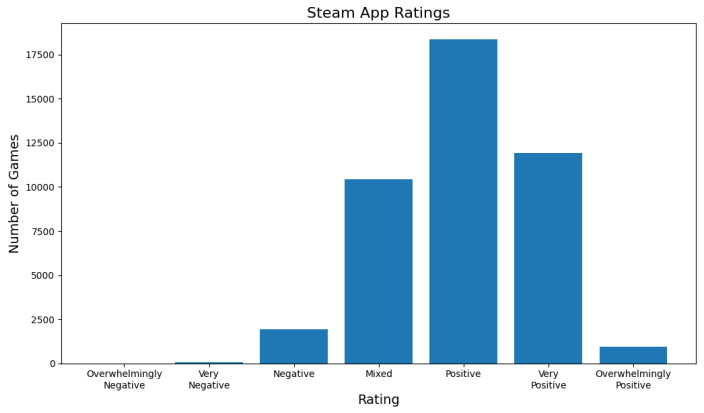

Fig. 8. Steam DLC Discounts

E. Genre & Description

The results of the proportions of the most-used words are interesting. There are clear differences between each genre in these figures. In figure 9, it is apparent that Racing, Strategy, Sports, and Simulation games use the word 'new' at a higher rate than other genres, with Racing being the

top. There are many possible reasons this could be the case. For racing and sports games, it may be important to the marketing strategy of these games to emphasize the new features that they have added to their game in order to keep it interesting to consumers, since these games tend to have the same gameplay most of the time. An interesting point to notice is that the genre Early Access has a disproportionately low use of this word. Since early access games are inherently new, and that is part of their novelty, the developers may not feel the need to emphasize the newness of the game or its features.

Results for the word 'play' can be seen in Figure 10. The genre that uses this word the most is Sports, with double the proportion of the second highest genre, Casual. This could be due to the marketing of sports games in particular, since they may want to emphasize the fact that the consumer is going to play their favorite sport and play as their favorite real-life players. Two interesting low-proportion genres are Education and Role Playing Game (RPG). Education's proportion makes sense, since the focus is on learning rather than gameplay. RPG's proportion also makes sense, but may not be intuitive. Since this is a role-playing genre, the emphasis may be on the experience of the player in the immersive story rather than the gameplay they will have.

The word 'world' has results that are intuitive, as shown in Figure 11. The Massively Multiplayer (MMO) genre uses the word much more often than any other genre. Since this game genre is all about interacting with other players in the world, the designer would want to emphasize this world and its features in the description. Likewise, RPG games are about world interaction, so these would emphasize the world for the same reason. An interesting data point is the proportion of this word under the Free to Play genre. Intuitively, this makes sense, since there are many free to play MMO's and RPG's on the market.

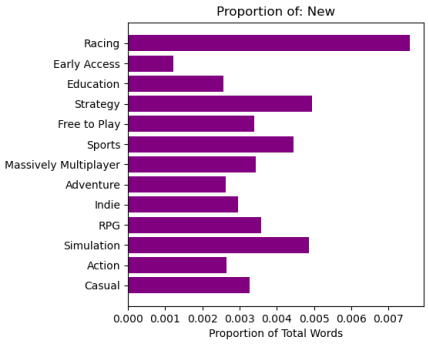

Fig. 9. Proportion of word 'New' in each Genre

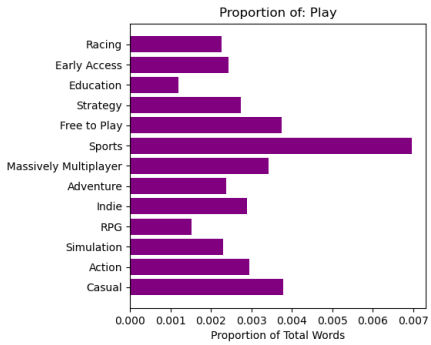

Fig. 10. Proportion of word 'Play' in each Genre

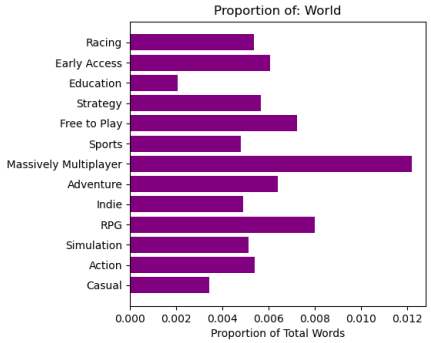

Fig. 11. Proportion of word 'World' in each Genre

The tagging results can be seen in Figures 12-14. The distribution of proportions for nouns (12) and adjectives (14) have no real pattern to them, however, verbs have some differences among the genres. It seems that the Casual genre uses verbs more than other genres. RPG, Early Access, and Strategy have the lowest proportions of all the genres. The low number for the RPG genre makes sense, since the descriptions for these games should be more about descriptive words, like adjectives. The reasons behind the results from Early Access and Strategy are not clear. These results have a caveat: if all of the verb tenses were taken into account, the graph may look more similar to the nouns and adjectives graphs, since this is only taking into account the present tense.

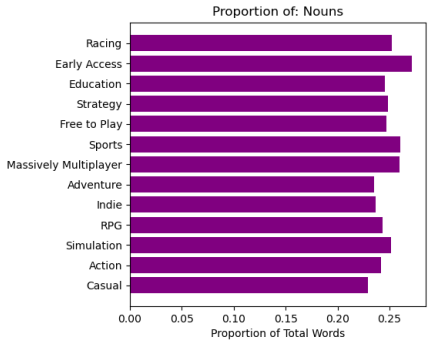

Fig. 12. Proportion of Nouns in each Genre

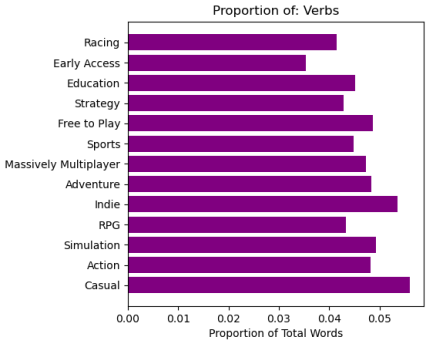

Fig. 13. Proportion of Verbs in each Genre

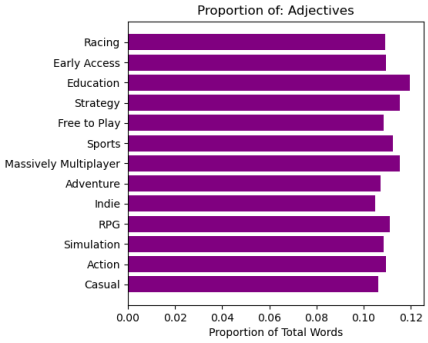

Fig. 14. Proportion of Adjectives in each Genre

F. Minimum and Recommended System Requirements

Due to what is explained in the data processing section, this task cannot be analyzed by statistics like other tasks. Below are two examples of results of system requirements. For the Operating System, it required Windows 7 at least with an i3 core and 4GB ram. Because this is a video game, it required the NVIDIA GTX 760 or AMD Radeon to make the graphics look better. The only things different on the minimum requirement system are processor and graphics. As the results, the minimum requirements are lower than the recommended system, this is quite understandable.

```
black squad - ea free timed weapon package 1,  654170
OS: Windows 7 64bit
Processor: Core i3-4170 or AMD FX-8300
Memory: 4 GB RAM
Graphics: NVIDIA GTX 760 or AMD Radeon HD 7950
DirectX: Version 9.0
Storage: 7 GB available space

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
black squad - ea free timed weapon package 2,  654171
Requires a 64-bit processor and operating system
OS: Windows 7 64bit
Processor: Core i3-4170 or AMD FX-8300
Memory: 4 GB RAM
Graphics: NVIDIA GTX 760 or AMD Radeon HD 7950
DirectX: Version 9.0
Storage: 7 GB available space
```

Fig. 15. Recommended System Requirements

```
black squad - ea free timed weapon package 1,  654170
OS: Windows 7 64bit
Processor: CORE2 DUO 2.2GHZ / AMD Athlon 64 X2 2.66GHz
Memory: 4 GB RAM
Graphics: NVIDIA GEFORCE 8600 OR GT630 / RADEON HD 6750
DirectX: Version 9.0
Storage: 7 GB available space

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
black squad - ea free timed weapon package 2,  654171
Requires a 64-bit processor and operating system
OS: Windows 7 64bit
Processor: CORE2 DUO 2.2GHZ / AMD Athlon 64 X2 2.66GHz
Memory: 4 GB RAM
Graphics: NVIDIA GEFORCE 8600 OR GT630 / RADEON HD 6750
DirectX: Version 9.0
Storage: 7 GB available space
```

Fig. 16. Minimum System Requirements

X. Problems Faced

Primary issues encountered during this project were related to collecting the data. The first issue was due to the HTML layout of steam store web pages not being universally organized. This made it difficult to consistently find targeted HTML elements containing needed information. Fortunately, most pages were consistent, but some were not leading to either data not being found at all or being incorrect.

The second issue was caused by requests timing out either due to bad appIDs or pinging Steam's domain too frequently, though bad appIDs was the primary reason. To counteract this problem, we simply added a try/except block to the request code to allow scripts to continue if a bad request was made.

The third issue was simply the fact that the runtime was over 22 hours for sequentially processing each appID in the list which consisted of roughly 130,000 apps. To shorten runtimes to durations suitable for one run, some of us used Python's multiprocessing capabilities to run multiple script instances essentially parallelizing the execution. Another solution was to simply add stop/resume functionality to their scripts allowing the user to stop execution whenever and resume where they had left off the next time the script was executed.

XI. Future Work

One important consideration for future work is to evaluate the important tags and simplify the tagging system in order to get more accurate results regarding the proportion of each word type. NLTK has a simplified tagging system built into it, but it requires more effort to implement, so there was not enough time to implement this in the project. This would give better results on word types like verbs, since the library splits up the verbs into different tenses (past, present, future) and tags them differently, so it is difficult to get an accurate count with this system.

Another important consideration is the scope of the items on the Steam store that was analyzed. For this project, the entirety of the Steam store was analyzed, which includes TV shows, photo editing software, and many more types of media other than games. To increase the accuracy of the

results, the analysis should be changed to filter out any non-game media.

The future work to implement would be the model to predict a game's genre based on its features. This was the original goal of the project if time permitted, but unfortunately there was not enough time to implement this feature. This would be a great way to further integrate the collected data.

XII. Organization

Start Date	Milestone
October 11	Determined how to get HTML from the steam store programmatically. Began looking at how to parse the HTML for data.
October 18	Use Steam API to gather preliminary data
October 25	.Discuss how to obtain data since Steam API did not give us much workable data
November 1	Scrape data directly from Steam
December 5	Analyze results and finish the final report and presentation.

Fig. 17. Timeline of the Project

Each team member was responsible for scraping data pertaining to a particular topic. Tom scraped data relating to genre, price. Matthew scraped data relating to DLC price/discounts and user reviews. Rob scraped data relating to descriptions. Tan scraped data relating to system requirements. After each team member scraped their data, they were then responsible for the analysis of their data. All team members worked together to write the final report and presentation.

The team regularly had weekly meetings to discuss the progress of the project. Each team member would let the others know their progress. The meetings were also used to discuss how data was going to be stored and formatted, how data was going to be collected, and how members were going to use their datasets.

References

[1] M. Honnibal, "A good part-of-speech tagger in about 200 lines of python," Available at https:// explosion.ai/blog/part-of-speech-pos-tagger-in-python (2021/12/09).