Steam Game Analytics

Nadav Hames
nadavh@my.yorku.ca
York University
Toronto, Ontario

Abdulfatai Adeshina
arules15@my.yorku.ca
York University
Toronto, Ontario

Duc Do
danny322@my.yorku.ca
York University
Toronto, Ontario

Karan Parva
karan26@my.yorku.ca
York University
Toronto, Ontario

ABSTRACT

It is no secret that the video game industry today is growing at a staggering rate and is worth more than the film and music industries combined in terms of revenue - generating upwards of 100 billion dollars per year globally. Companies looking to cash in have many important decisions to make before developing a game and entering into the market. These decisions can make or break the success of a game and include selecting a genre, price, engine and platforms to support. In this decision making process, it can be unclear which choices will bring positive, profitable results after only a manual, surface-level inspection of current trends. Our solution uses Apache Spark amongst other tools to aggregate data and present time series results and other metrics in real time. This data is sourced from relevant gaming platforms and news outlets including the Steam Store, Youtube, and Pcgamer. This approach aims to provide accurate analytics from which companies can derive insights and make informed decisions.

ACM Reference Format:
Nadav Hames, Abdulfatai Adeshina, Duc Do, and Karan Parva. 2019. Steam Game Analytics. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Our project attempts to provide a platform to enable the extraction of useful analytics from a vast, growing landscape of tens of thousands of video games spanning many genres, prices and platforms. Companies looking to enter into the lucrative video game marketplace can use our platform to make data-driven decisions about what type of game they intend to develop in order to succeed. We do this by providing a scalable method of continuously extracting, transforming and aggregating relevant data in real time from many game platforms, news websites, and the like.

A myriad of questions can be asked on this intricate topic, primarily revolving around the question of why do people enjoy the current selection of popular games? Is it because people like a certain genre more than others? Or could it possibly be attributed to a given game being free or paid that affects its appeal? These are just some simple questions, but more complex ones can be asked to allow developers to understand why a video game performs well on gaming platforms like Steam. For example, it might be useful to investigate trends relating to when certain developers publish new games and to check whether that developer has published any games previously. From this information we can find out if the developer's previous games are having a significant impact on the popularity of his/her new game. If a significant correlation is found, this would mean that publishing many games on Steam helps boost the popularity and therefore profitability of a developer's video games.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

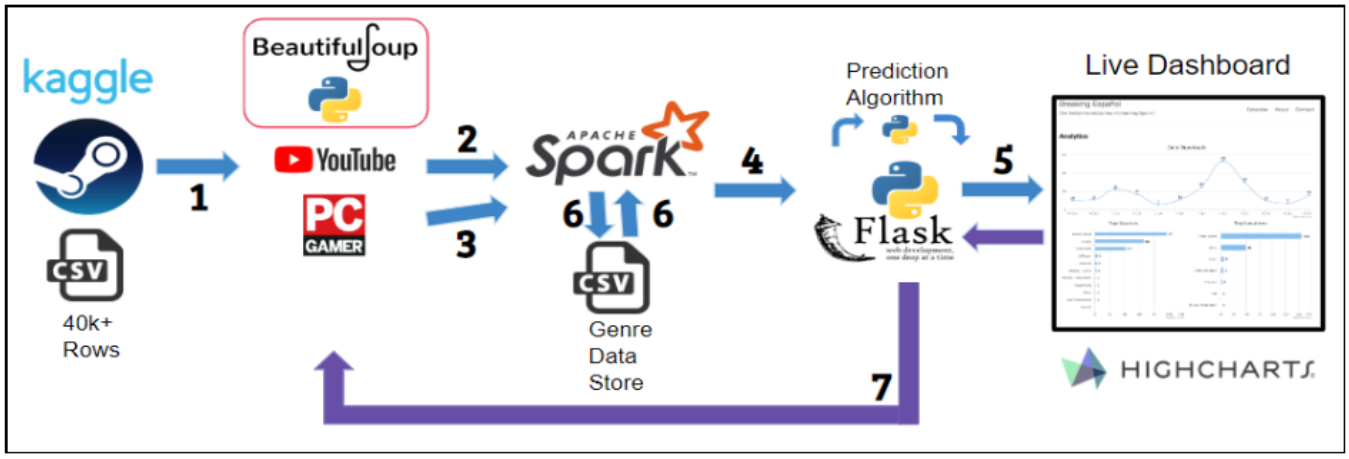
2 DATA AND DATA ANALYSIS

2.1 Data Dimensions

Aspect	Dimension	Value
Data	a) Structure	Semi-Structured
	b) Size	Medium
	c) Sink Rate	High
	d) Source Rate	High
	e) Quality	High
	f) Completeness	Complete
Processing	g) Query Selectivity	High
	h) Query Execution Time	Long
	i) Aggregation	Medium
	j) Processing Time	Short
	k) Join	None
	l) Precision	Approximate

a) The scraper scrapes raw data from youtube and pcgamer.com and converts it to JSON to pass onto Spark

b) The amount of youtube videos to be scraped can be changed but for the purposes of this paper has been set to 200 videos per game, chosen with the same criteria used to return search results on youtube, our steam dataset contains over 40k games,therefore a full analysis can expect to generate and process $40,000 * 200$ json objects which is roughly 8 million JSON objects, if we assume that each object is about a Byte in size (in reality they are probably larger) the total size of the youtube data processed is about 8GB. As for the pcgamer.com dimension, the word count for each of the 40k games is computed for each pc gamer article this year (between 1k and



10k) determining the word count is a relatively cheap operation and majority of the time the word count for a game is 0, so on average we can say a game will appear in 25 – 100 articles, multiply this by the amount of games we have and we have a number of useful JSON objects generated at about 2-5 million, using the same estimation technique as before we come to about 2-7 GB of data processed.

c) The scraper makes multiple updates per second

d) The Data is sent to spark, aggregated, and displayed live as data comes in

e) A few bad eggs of data will not mess the results up significantly

f) The system will crash if some JSON data are missing, thus the incoming data must be complete

g) Users have the ability to control selectivity, but generally <20% of data will be required for an analysis

h) We predict on a distributed system with direct access to the Youtube API execution times can be shortened, but for now the scraper must manually scrape data from Youtube which takes quite a bit of time

i) Aggregations such as weighted sentiment averages based on article word count were performed

l) The results are mostly accurate but regarding Youtube, our system may encounter outlier videos which may skew results or videos which are not related to the game but somehow show up in the results - likewise with Pcgamer articles

2.2 Preprocessing, Data Cleaning and Data Transformation Steps

A few preprocessing steps are taken by the web scrapers before data is sent to be processed. Date information is inconsistent between sources so the Python datetime library is used to transform it into month/year format. For each game in the Steam dataset, there can be multiple genres, so these genres are exploded outwards so they can be accurately processed.

2.3 Analysis

Our solution involves taking weighted averages of different metrics over genres per month. To obtain an accurate sentiment score representative of all considered games, we have come up with a formula to weigh game sentiment per article based on the word count of the game in the article. For each game:

```
game sentiment += (game word count in
article) * article sentiment,
```

and using sparks aggregate key method we can keep track of the total word count and at the end of the job, we'll have a list of the following format:

> [Jan: [sum of sentiments for Jan, wc for articles in Jan], Feb: [sum >of sentiments in Feb, wc for articles in Feb] ……]

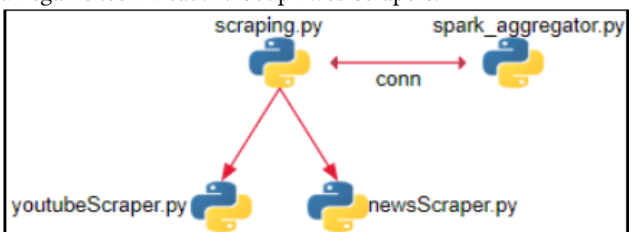
The last step is to divide each sentiment value by word count to obtain the weighted sentiment value to be sent to the frontend, this formula ensures that games mentioned more frequently in an article have a greater contribution to the sentiment value contributed by that article, for example if CS:GO is mentioned 20 times in an article and Minecraft is mentioned once, it would be inaccurate to say that CS:GO and Minecraft contribute the same value to the article sentiment, with our formula CS:GO will contribute 20 times more to the articles sentiment then Minecraft would. We have also provided some other graphs useful for analytics such as total youtube views per genre over time and total word count in all pc gamer articles per genre over time.

3 ARCHITECTURE

The architecture of our project is roughly inspired by and builds upon Assignment 3 where the goal was to implement an architecture to provide streaming text analytics. It features an ingestion process that collects data from both the Kaggle Steam dataset and two web scraping historical streaming sources (Youtube and Pcgamer.com) with the ability to expand to more sources in the future. The data is then immediately processed, aggregated and stored by Apache Spark before being served via a Flask server to a live dashboard with visualizations. Throughout the process, the data is kept in json format. Included above is a diagram that shows the architecture together with the data flow - an in depth description of which is provided below: (The numbers in bold correspond with numbers 1 through 7 on the diagram)

3.1 Data Ingestion

The process starts with the Kaggle Steam dataset. It contains approximately 40k rows with each row containing information on a single game or piece of software sold on Steam. **1)** The game rows from the Steam dataset are read sequentially by both the Youtube and Pcgamer.com BeautifulSoup4 web scrapers.



These web scrapers execute simultaneously on two separate processes in a Python Docker container. Both scraper processes are created by a parent process which passes down the connection it establishes with Spark. For each game, the Youtube scraper visits a user specified number of relevant videos and sends data of the following fields to spark:

2) <month,year,genre,game,likes,dislikes,views>.

A set of previously scraped video ids are saved by the scraper to ensure that videos are only scraped once. At the same time, the Pcgamer scraper receives a range of months from the user and for each game in the Steam dataset it scrapes all articles from each month in the range and appends them to a csv file. It then checks the word count of the game in question and overall sentiment (using the Textblob library) in each article and sends data of this form to Spark:

3) <month,year,genre,game,wordcount,sentiment>.

3.2 Data Processing and Data Storage

An Apache Spark script running in another docker container (or locally) is then used to aggregate the incoming data from both scrapers. The data is averaged and weighted where applicable so that results are over entire genres per month, not just individual games. A description for our sentiment weighting formula has been given above, The resultant data is periodically saved to a file by Spark**6)** so that there is historical data the next time the platform is run from which to continue aggregating. The data is then sent to a Flask server in the following formats based on the aggregation being performed:

4) <genre,year,month,[word count, sentiment]>

for the sentiment and word count plots, and

<genre, year, month, total Youtube views>

for the youtube views over time plot missing data

3.3 Data Serving

Both the Apache Spark Service and the front end dashboard are connected to each other using a Flask server. The Spark job sends POST requests with new data to the Flask server every couple of seconds. Each Plot has its own API endpoint on Flask to which it sends data to, This data is stored on the server and converted to the proper format to served to the NVD3 charts, which sends GET requests to the Flask server every 5 seconds polling for new data to dynamically update the charts.

3.4 Data Visualization

To display results in an interactive, responsive and interesting way, several NVD3 charts are displayed to the user in a dashboard. These charts update as new results come in from Spark via the Flask server. The data received is in the following format:

5) <[genre:[date:weighted sentiment]]>

for the sentiment plot:

<[genre:[date:word count]]>

for the word count over time plot, and

<[genre:[date:youtube views]]>

for the youtube views plot.

3.5 User Specified Parameters

7) Currently users can specify parameters to be used to select data such as Genre from the scraping.py date ranges, and a cap on the amount of videos to be scraped for each pass which gives users the ability to make a tradeoff between query execution time and accuracy of data. This feature is a work in progress and is something we would like to provide in the future.

3.6 Architecture Limitations

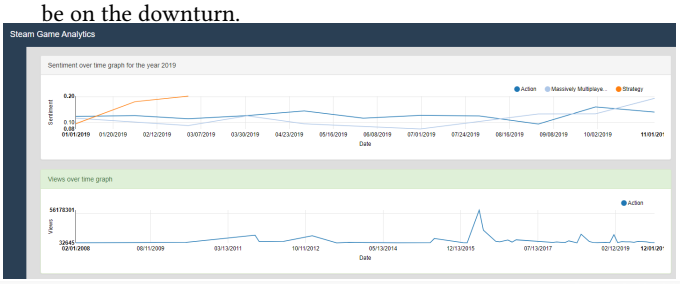
- As scraping the web is a relatively slow process, this means that building a sufficiently sized dataset to perform Big Data analytics would take an unreasonably long period of time. To overcome this limitation, the data is scraped from the web and then is immediately sent as a stream to be processed and displayed. This means that the longer duration the pipeline executes, the more accurate the results. This could be another limitation as high accuracy/resolution results are therefore not immediate.
- Our project is reliant on web scrapers which is slow, and requires separate lengthy implementations for each platform. This could be overcome if API access is provided with no data cap.
- The Steam dataset we use as our source of games is limited to games on Steam. This can be remedied in the future by combining this dataset with other datasets from different game platforms like the Epic Games store or the Windows store.
- One inefficiency that exists in our architecture is that for each game, the word count for that game is calculated for every Pcgamer article, it might be quicker to have a batch job precompute the word count for every game in each article, and have it stored in a database or csv file to avoid frequent and inefficient computations and speed up the overall analysis process

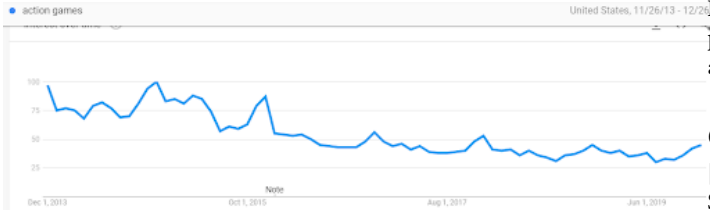
4 EVALUATION AND RESULTS

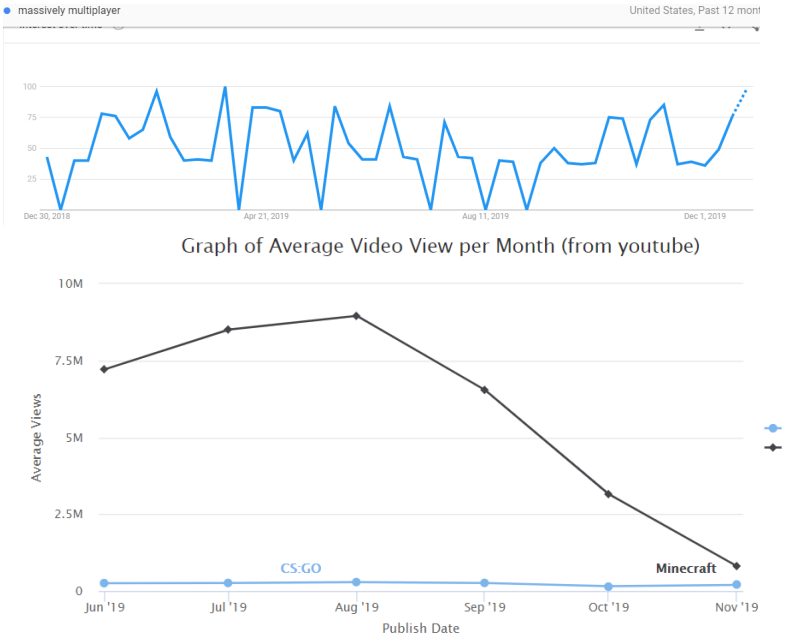
Throughout the development process, we used various methods to evaluate and test our work. We were able to test our architecture by scraping Youtube data, Pcgamer articles, and data from our steam

dataset, all which were joined together to produce results shown below

- We found that graphs roughly align with Google search trend data, but provide more fine grained detailed results
- For example, the youtube views plot tends to align more with google trends data for data points with a more recent release date, this is because the Youtube search results are biased to returning newer videos.
- As a result, the MMORPG chat trends in relatively the same directions as the google trends plot the nearer we get to the current date
- As can be seen in both our Action game plots and the Google trends Action game plots, there is a spike in both plots towards the end of 2015, this may tell us that there might have been some event on youtube which caused a surge in interest in the game around this period
- We can also see the influence that one or 2 key influencers may have on the overall popularity of a game from our analysis
- Shown below is a plot of total Minecraft views for the months between June and November 2019, there is a peak in August which can be explained by the fact that a month prior, popular youtuber pewdiepie decided to start his minecraft series on youtube.
- We can see as the fall months approach, there is a steep decline in the views which is explained by the fact that pewdiepie's minecraft series had ended, as well as the fact that school in western countries restarts in september, affording children, the main target demographic of minecraft, less and less time to spend playing the game and exploring videos of the game
- By examining such sharp peaks and troughs amongst the many graphs our architecture has the potential to generate, we can find a good variety of anomalies which may offer a game producer an opportunity to enter a hot market before everyone else or pull the plug on a market which seems to be on the downturn.







5 CONCLUSIONS

Our Data Architecture provides a useful pipeline for Data Driven Organizations to evaluate various gaming markets. The Architecture is built to easily scale to the needs of a large organization (for example, the YouTube scraping portion would translate very nicely to a distributed environment where each node is assigned a portion of the scraping job, data can also easily be stored for future access, and as obvious, the usage of spark lends itself nicely to distributed environments) As well as making it very easy to add new features without changing much of the existing codebase, the data being fed into spark is highly structured and is very rich providing programmers and data engineers easy access to interesting raw data to be used to their desire, it provides a plug and play architecture for fast moving Data Driven Organizations whose needs are ever changing and in which its necessary to act fast to determine product market fit for planned projects. Unfortunately our analytics were rather limited but there are a host of interesting analytics that can easily be integrated into our architecture (eg. pie charts of overall youtube market capture for each genre, bar charts for year on year growth for gaming segments etc. A prediction algorithm which extrapolates the next month of values is shown in our architecture diagram and is something that can be quickly implemented at the Flask level.) In summary, we have built our Steam Games Analytics project to be a useful, effective tool for data-driven companies to add to their analytics arsenal.

6 REFERENCES

[1] "Apache Spark™ - Unified Analytics Engine for Big Data," Apache Spark™ - Unified Analytics Engine for Big Data. [Online]. Available: https://spark.apache.org/. [Accessed: 26-Dec-2019].

[2] "Build software better, together," GitHub. [Online]. Available: https://github.com/. [Accessed: 26-Dec-2019].

[3] "Docker Hub," Docker Hub. [Online]. Available:

https://hub.docker.com/. [Accessed: 26-Dec-2019].

[4] "Flask," Full Stack Python. [Online]. Available: https://www.fullstackpython.com/flask.html. [Accessed: 26-Dec-2019].

[5] Hax, "Steam games complete dataset," Kaggle, 16-Jun-2019. [Online]. Available: https://www.kaggle.com/trolukovich/steam-games-complete-dataset. [Accessed: 26-Dec-2019].

[6] JS Foundation, jQuery, 2018. [Online]. Available: https://jquery.com/. [Accessed: 26-Dec-2019].

[7] Novus Partners, "NVD3 Re-usable charts for d3.js," NVD3. [Online]. Available: http://nvd3.org/. [Accessed: 26-Dec-2019].

[8] M. Otto and J. Thornton, "Bootstrap," · The most popular HTML, CSS, and JS library in the world. [Online]. Available: https://getbootstrap.com/. [Accessed: 26-Dec-2019].

[9] "PC Gamer," All Content Archive | December 2019 |PC Gamer. [Online]. Available: https://www.pcgamer.com/archive/. [Accessed: 26-Dec-2019].

[10] T. Pechlivanoglou, "Intro to Apache Spark." [Online]. Available: https://www.eecs.yorku.ca/ papaggel/courses/eecs4415/docs/tutorials/tut8-apache-spark.pdf.

[11] Pypa, "pypa/pipenv," GitHub, 24-Dec-2019. [Online]. Available: https://github.com/pypa/pipenv. [Accessed: 26-Dec-2019].

[12] L. Richardson, "Beautiful Soup," Beautiful Soup: We called him Tortoise because he taught us., 24-Dec-2019. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/. [Accessed: 26-Dec-2019].

[13] "Streaming Text Analytics using Python," Nov-2019. [Online]. Available: https://www.eecs.yorku.ca/ papaggel/courses/eecs4415/docs/assig
YouTube. [Online]. Available: https://www.youtube.com/. [Accessed: 26-Dec-2019].