Predicting Video Game Popularity With Tweets

Casey Cabrales (caseycab), Helen Fang (hfang9)

December 10,2015

Task Definition

Given a set of Twitter tweets from a given day, we want to determine the peak number of people that are playing a video game that day. When blockbuster games are released, thousands of people on twitter are talking about it by posting impressions, pictures, and videos. The excitement for the release is also shown in the number of users playing that video game. But as the release date drifts further away, tweet volume and sentiment die down as people finish the main story or stop playing altogether for one reason or another. So our goal with this project is to predict the correlation between features of tweets about a certain video game and the number of players who played it that day and moving forward.

Our system takes as input a database of tweets separated by game, including Grand Theft Auto V (2574), Counter Strike: Global Offensive (3495), Skyrim (1201), Rocket League (556), The Witcher 3 (165), and Trove (506). We chose these games for multiple reasons: time distance from release (Skyrim), major events occurring about the game (Counter Strike), extremely popular games (GTA), mid-tier release games (Witcher 3), and smaller indie games (Trove). These are filtered from Tweets from the months of July and August of 2015. As output, the system returns the predicted peak players on that day based on Tweet features which are then compared to actual Steam player data for the days of those months to measure our accuracy. We display a summary of these results for each game as well as error analysis statistics.

In order to evaluate our system, we are comparing the number output by our model to the number of people that played according to the data we found from Steam. This constitutes the error from the hard data points over each day of the two months.

The information output can be useful for both the developer and player-base (as well as interested consumers). Developers can track what influences player trends for their game, whether it be esports competitions, DLC releases, or bad updates. These trends can influence when they decide to release new content so that on the downturn of players, they can get the number back up where they want it. The player-base can use these predictions to gauge whether online play will be competitive, or judge how popular the game is at a given

time. Executives can use this information to make key marketing decisions. The twitter data should be able to predict the player trends of a video game into the future based on existing data.

Infrastructure

This section will outline the type of data we used, as well as go into more depth about how we acquired and processed it to meet our needs. Our data is made up of Tweets from a given day and game. We downloaded archived, raw, unprocessed Tweet data for the months of July and August, 2015 from archive.org. It was in no way filtered or marked, and was simply the raw tweets themselves with the metadata provided by Twitter. Because the Twitter API doesn't allow us to stream Tweets from far enough back in time, we had to use this resource which contained JSON grabs each day from the 'Spritzer' version of the Twitter stream. In total, it contained about 900GB of Tweets over the entire two months. We also began to download and archive our own tweets for the end of November, but decided that the more complete months would be more useful to the final result over more scattered data.

Once we had this data, we filtered it by crawling through the entire file structure and using regexes to look for occurrences of the different hashtags for our selected games (ignoring deleted Tweets). If we found one, we would save it to the respective output file for that game. We did not separate the data into two train/test files so that we could implement K-folding.

After parsing the data into one large file, we could use that to perform sentiment analysis and extract features for each game from each day. This portion was accomplished on a game-by-game basis, by viewing each tweet one at a time and creating the feature vectors (addressed later in this document).

On top of obtaining Tweets, we also had to find and format player counts for each day, for each game, in both July and August. This data was collected from SteamDB, which collects a variety of different statistics for each game. We were only interested in the player data, though. We recorded this information and placed into a JSON file separated by game, containing the game name, release date, and peak player dictionary for each day of July and August. This data is used for determining accuracy of our model and being able to train it well with the model.

In order to implement sentiment analysis, we also downloaded the Stanford Twitter sentiment corpus (**http://help.sentiment140.com**)[4], which consists of a training set of of 1.6 million tweets labelled based on emoticons ("4" being positive, "2" being neutral, and "0" being negative). While labelling tweets based on the presence of emoticons seems very cursory, this was the best and largest available labelled dataset that wasn't pigeoned to Tweets of non-game related topics. If we were given more time, we potentially would have explored hand-labeling tweets.

Approach & Error Analysis

The major challenges of building the system were data collection and determining useful features. With regards to data collection, it is difficult to get Tweet archives because of Twitter's terms of use. In addition, we can only use the API to download Tweets from the past two weeks, which is too short of a time span to glean any meaningful information from the dataset. So after spending a great deal of time searching for places to get unaltered Tweets, we finally found the archived data sets that we are using now. The second challenge of determining useful features also directly relates to identifying and capturing the phenomena in the data. Our goal was to determine how many people would play a game on a given day based on the Tweets about that game, so we needed to select features that were both general and useful enough to predict for all games. Having the metadata associated with each Tweet made this task a little easier, but it was still important to tune the features for the best possible results.

In order to predict the peak number of players per day for a given game based on a collection of tweets, we first looked at different pure regression models. In the preliminary phase of development, we looked at using linear regression.

Linear regression models the relationship between some input variable X with an output variable Y (also referred to as the response) such that:
$f_w(x) = w \cdot \phi(x)$, where w represents the weights (or coefficients of the respective feature). It is simple to implement and does a good job at approximating the models. However, regression depends heavily on the features we come up with, which could greatly alter any results we get with different feature combinations. For our baseline, we use a simple linear regression, where the input variable X represents a single feature - the number of tweets of the game per day. From there we then looked at multivariate linear regression, where the input variable X represents multiple features.

Number of tweets	50
Number of users	45
Avg Favorites	0.0
Avg Followers	124.11
Avg Retweets	0.0

$Value = 46956$

From the data we extracted, we decided to use features such as the number of tweets on the game, number of users tweeting about the game, average number of followers of users tweeting about the game, average number of favorites of these tweets, and the average number of retweets amongst these tweets (above is an example feature vector from GTAV). Taking these features from the training data, and their corresponding peak number of players, we proceed to "fit" the data into the model by using least squares.

With least squares, we want to find the w (weight) that minimizes the objective function:
$TrainLoss(w) = \frac{1}{|D_{train}|} \sum_{(x,y) \epsilon D_{train}}^{\cdot} (w \cdot \phi(x) - y)^2$

In order to determine the value of w, we perform a stochastic gradient descent on the $TrainLoss$ objective function using the training data (features and values) such that for each training data, we tune w until it converges as follows:

$w \leftarrow w - \eta \bigtriangledown_w TrainLoss(w)$

After running the entire training set for a given game through stochastic gradient descent, we take the learned value of w and evaluate it on the test set. The learned w will give us a predicted value which we then compare to the true value.

To illustrate the process, we use the elements from the training data (am example is shown above) to find w. For Trove, we have $[3.98411785e{+}01, 2.10001403e{-}14, 1.24102394e+03, 6.05673308e+02, 4.43222834e+01]$.

In our test set, we have a data point with the feature $[6, 28, 0.0, 394.85714285714283, 0.0]$. Using our linear regression model $f_w(x) = w \cdot \phi(x)$, where x is the feature vector and w is the weights, we estimate that the peak number of players for that day is 239437.801381, which is quite close to the true value of 24941.

In addition to a simple linear regression, we also implemented sentiment analysis to draw more meaningful features about the content of the tweets. In particular, the feature we developed is the ratio of positive tweets to total number of tweets. However, in order to label the tweets in our training and test sets with the appropriate sentiment, we developed and trained a classifier using the NLTK library.

In implementing sentiment analysis, we trained our classifier on the Stanford Twitter Sentiment dataset by doing the following:

- Extract word features from our tweets (all space separated words in lower case form with hashtags eliminated)
- Apply features to our Naive Bayes classifier
- Train the classifier with our labelled training sets

The Naive Bayes classifier which our nltk function is based on is a probabilistic classifier based on using Bayes theorem and assuming independence between the features. Essentially in labelling each tweet, the classifier is labelling it by selecting the class (positive, neutral, negative) that maximizes:
$\hat{y} = argmax_{k \epsilon \{1,...K\}} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$
where $\hat{y}$ and C_k are the class and x are the features.

Initial Approach Our initial approach as we iterated through the project involved splitting Tweets (only from August) into two data sets by date: Tweets made August 1-10 were treated as the training set, while Tweets through the end of August were used as the test set. However, we soon realized that our results were being skewed by the presence of abnormal events or other such occurrences between the training and test set dates.

Revised Approach In the final iteration of this project, we decided to implement K-folding. Overall our data set isn't very large, so to make up

for this the different combinations of K-folding make up for it. This method generalizes better to the dataset at hand than a 70-30 split for training and test sets. In addition, it combines measures of fit to correct for training error and derive a more accurate estimate of model prediction performance.

Baseline For our baseline, we use a simple linear regression with a single feature - the number of tweets of the game per day. It seems reasonable to conclude that games with a large active player base are more likely to get tweeted about that those with a smaller active player base. Using our initial approach of splitting the data two ways, with the training set as the set of daily tweets from August 1, 2015 to August 10, 2015 and the test set as the set of daily tweets from August 11, 2015 to August 31, 2015. We obtained the following results:

Game	Average Percent Error
CounterStrike	62.8117
GTA V	46.4884
Witcher3	1.8304
Skyrim	5.7772
Trove	8.9124
RocketLeague	6.7847

Using the K-folding technique described above this time with the baseline, we obtain the following results:

Game	Average Percent Error
CounterStrike	45.2570
GTA V	62.5597
Witcher3	2.0928
Skyrim	8.8263
Trove	659.8729
RocketLeague	2468.0833

Trove and Rocket League see a spike in error, which is likely due to their very recent mid-July release, which may display much more exponentially decaying figures compared to more mature games.

Oracle Our oracle essentially "cheats" and peeks at the actual peak number of players for a given game on a given day based on data provided by SteamSpy.

Multivariate Linear Classification Given a set of daily tweets, there are other features beyond the number of tweets about the video game that are important in determining the number of people playing a game. We enhanced our linear regression model to also include these features, based on the idea that how influential a twitter user is and how well-received the tweet is could indicate how many people play the game:

- Number of users tweeting about the game (Not necessarily equal to the number of tweets on the subject)
- Average number of favorites of tweets
- Average number of followers of the users tweeting about the game

- Average number of times gaming tweets are retweeted

Using the same training and test sets described in our baseline, we obtained the following results:

Game	Average Percent Error
CounterStrike	99.575
GTA V	1.20733
Witcher3	5.9382
Skyrim	90.821
Trove	1.1351
RocketLeague	1.2864

It is unclear whether or not multivariable linear regression outperforms our baseline regression. Games like Counterstrike and Skyrim suffer from extreme error, while other games showed a reduction in error. There are many factors that account for this variability including overfitting of data. From examining the data more closely, games like Skyrim have unusual player patterns in that peak number of players mapped over time looks like a sinusoidal curve; it is possible that this has to do with gameplay and may not be best expressed through linear regression. In addition, our data is sparse in that our training set only covers 10 days. With more data covering a more extensive period of time, it is possible that there would be less overfitting and the linear regression model could perform better.

Results With K-Folding

Game	Average Percent Error
CounterStrike	1.0406
GTA V	1.0442
Witcher3	1.6920
Skyrim	0.9405
Trove	1785.7114
RocketLeague	10.0558

It is apparent that using K-Folding outperforms the traditional split of training and test data. Counter Strike error is completely reduced, because of how we eliminate bias from the World Championship beginning and skewing the data for that time period. Games with smaller Tweet datasets also saw some improvement from our algorithm being able to get a better feel for the data itself. However, because the release dates of Trove and Rocket League were toward the beginning of July, their errors are affected and can lead to extraneous prediction results on the days prior to release. By using K-Folding we help prevent overfitting to the data at hand.

Final Results of Linear Regression with K-Folding and Sentiment Analysis (ratio of positive tweets to total tweets)

Game	Average Percent Error
CounterStrike	0.3547
GTA V	0.9393
Witcher3	1.0318
Skyrim	1.0003
Trove	45.6576
RocketLeague	1390.2450

The final step in our process was the addition of sentiment analysis to help predict number of players. This feature gave some dimension to the data at hand by telling us what the people are thinking when they post their Tweets. So the games with larger datasets definitely get more accurate because of this method. However, sentiment of the smaller games affect the result a lot more and can get skewed more easily, so the average error is slightly increased. It also appears that games with release dates during the time of our data are still negatively affected. Additionally, the labelled training set we used to train our Naive Bayes Classifier may not have been the most optimal because the 1.6 million tweets were labelled based on emoticons (which are the not the best indicators for sentiment) and these tweets came from a wide variety of sources. For future evaluation, we would explore hand labelling gaming tweets because the vocabulary used and the nature of gaming tweets differs from the larger community.

Extending Features

In addition to our existing set of features, we extended these features to see if there are non-linearities in the features. And if so, would these extensions improve results?

After experimenting with some combinations, it is unclear whether or not these extensions would improve the results of our regression model. Often in times, the average percent error from the games Trove and Rocket League would swap in having egregiously large errors or the other games would experience slight increases or decreases in error. For example, when we added the feature $numTweets^2$, it appears that for games like Rocket League, tweets and players are more quadratically related:

Game	Average Percent Error
CounterStrike	1.002
GTA V	1.003
Witcher3	0.825
Skyrim	1.491
Trove	153.318
RocketLeague	54.677

We also experimented further. It intuitively makes sense that the average number of favorites on a tweet is related to the average number of followers. We tried the cross term $followAvg * favAvg$ and yielded the following results:

Game	Average Percent Error
CounterStrike	1.0019
GTA V	1.0010
Witcher3	0.6579
Skyrim	1.3948
Trove	306.0377
RocketLeague	2.7019

Overall this term yields better results in that RocketLeague's error decreased dramatically but most other games increased in error. It is likely that there is a significant relationship between the number of followers and the number of favorites that is displayed in RocketLeague's tweets, but much less so in Trove.

Literature Review

We did not find another system that also tried to predict player population for a game on a given day using Tweets. However, there is a report that investigates using Twitter to predict the stock market titled "Twitter Mood Predicts the Stock Market."[2] This report is similar to what we are trying to do with video game prediction, and while not exactly the same, is comparable and complementary to our work.

Similar to our implementation, Tweets are analyzed by their text through sentiment analysis on a given day to assess positive or negative public mood. This provides a dimension for evaluating the relationship between number of Tweets and what their meanings actually were. We also read in the stock market prediction report that they used K-folding cross-validation to better improve upon the accuracy of the sentiment. So we also decided to implement this, as it was a way for accounting for extraneous events that may have skewed the results otherwise. Whereas they accounted for major holidays and Election Day, for example, we accounted for E-Sports events and launch days of video games. This technique ended up drastically reducing the overall error for our games by mirroring their tactics.

However, there are some important differences. The mentioned report only utilizes sentiment analysis for their prediction, with 7 mood dimensions generated from two different tools. Our prediction of player populations is a combination of a regression model and sentiment analysis so that we can best fit the relationship of the pieces of data and generate the best possible prediction. They do mention that a single dimension may ignore the multi-dimensional structure of human mood, but in order to compensate for that in our study we applied regression with additional features instead of adding additional mood dimensions. This helped to take a holistic approach of the data.

References

1. Internet Archive Search Tweets

2. Bollen, J., Mao, H. and Zeng, X.-J. 2010. Twitter mood predicts the stock market. Journal of Computational Science 2(1):1-8.

3. Luce, Laurent. 2 Jan. 2012. "Twitter Sentiment Analysis Using Python and NLTK.Laurent Luce's Blog

4. Stanford Twitter Sentiment Sentiment140