Text Mining on Android Game App Review

Prateek Mohta
Nikhar Khandelwal

Table of Contents

1. Problem Statement

As smart phone has become an essential part of our lives, the commercial value and potential profits we can gain from mobile applications is tremendous. However, developing a popular app is not an easy task, considering the intense competition in app market due to the low barriers of entry for app development. Therefore, it is vital for app developers to receive and respond timely to user feedback. App stores usually provide systems for users to give ratings and comments on the apps, and we believe user comments are valuable information for app developers to improve user experience and satisfaction. Additionally, we feel Twitter feeds also play a pivotal role in determining user sentiments.

Due to the large volume and textual nature of user reviews, it could be challenging for app developers to efficiently extract information from it. In this project, based on the data collected from Google Play Store and Twitter, we will use text mining techniques such as sentiment analysis and term frequency analysis to analyze the user reviews, and identify useful insights for app developers, such as key features of the games that users like the most and the weaknesses which need to be improved. We hope our findings could help improve the efficiency and effectiveness of the app store feedback system, and help app developers to better understand and meet the user expectation.

2. Datasets

2.1 Source

We have chosen Google Play Store (https://play.google.com) and Twitter as our data source.

With almost 3.5 million apps available for download, as per research carried out by Statista.com in 2017[1], 9 out of top 10 leading apps in terms of revenue were gaming apps. This led us to choose the most popular games as our focus for data collection in form of user comments and ratings as a major chunk of overall revenue from Google Play Store is concentrated there.

Since most people nowadays prefer to voice their opinions on social platforms like Twitter, we also decided to fetch tweets related to the selected apps and combine the datasets.

2.2 Collection Methods

The data collection process is performed in two stages-

(1) Due to the lack of resources to process huge amounts of text data, only popular apps for adventure category, approx. 100 apps, were selected from Google Play Store.

[1] Source: https://www.statista.com/statistics/271674/top-apps-in-google-play-by-revenue/

(2) For every application selected in the first stage, reviews and tweets were collected from the date of app release or for the last 7 years, whichever is closest.

In the absence of any free public API from Google for its play store, all the relevant data has been collected by scraping the dynamic Google Play Store web pages for each android app. The reviews were first sorted by 'Most Relevant First' before copying them.

Following tools will be used to gather and assemble the data from play store:

- Selenium python library to simulate mouse clicks and page scrolls at random intervals on the play store application's web page. This is required as the Play Store web page loads dynamically.
- Google Webdriver binary used by the selenium to simulate the process of accessing the web pages, navigating and scrolling through the reviews section.
- BeautifulSoup python library to parse the web pages and extract useful data from the HTML source.
- Twitterscraper[2] script to scrape for Tweets using the Python package by retrieving the content for the required hashtags and using Beautifullsoup4 to parse the retrieved content.

2.3 Variables

The key variables of app reviews and comments scraped from the Google Play Store are summarized in **Table 1** as below:

Variables	Type	Explanation	Example
Application Id	String	Unique identifier for the selected application whose reviews are being scraped.	com.nianticlabs.pokemongo
Review Rating	Integer	The number of stars given by a reviewer to the application.	5
Review Comment	String	Feedback given by the reviewer in text format.	Wow.... Love this game. My son and I get to play together, AND it's one thing that gets him out of the house.
Review Posting Date	Date	The date on which the comment was posted.	19-Jan-18

Table 1: Summary of Key Variables in App Review Dataset

The key variables for the data retrieved from Twitter for all apps in the selected list are summarized in **Table 2** as below:

[2] Source: https://github.com/taspinar/twitterscraper

Variables	Type	Explanation	Example
Tweet Text	String	Textual data provided by the twitter user in a tweet.	I love Pokemon GO lol waiting for that 2nd Gen tho. The best Gen
Likes	Integer	The number of likes received from the viewers of this tweet.	1
Tweet Date	Date	The date on which the tweet was posted.	05-Jan-2018

Table 2: Summary of Key Variables in Twitter Dataset

2.4 Pre-processing of data

Data cleansing, though tedious, is perhaps the most important step in text analysis. Dirty data can play havoc with the results. Furthermore, data cleaning is invariably an iterative process as there are always problems that are overlooked the first time around.

Therefore, before processing the data and building models to identify important features from the app reviews, the data was passed through certain filters to ensure the reviews collected are not just plain noise and can provide some meaningful insights. Some of the filters that were applied on the data were:

- Apps found to have less than 3000 reviews were discarded from the model as they tend to have a lot of non-relevant and redundant text.
- Any links in the review or tweet text were removed.
- Numbers, punctuations and stop words based on a custom stop words dictionary, were also removed from the texts.
- Stemming and lemmatization of the text.
- Tweets were subjected to another filter where only those tweets were considered that had at least one like, reply or retweet. This was to ensure that the tweet was actually a review and not some promotional tweet from the game handle or just some achievement/score tweets made by the game user.

3. Data Analysis

3.1 Overview of Datasets

After removing apps with less than 3000 reviews, there are 53 apps left in the final dataset, which are listed in **Appendix A**. The dataset consists of 363,129 rows with 6 variables, namely App Title, Date, Rating, Review Helpful, Review Text and User Name. For tweets dataset, 1000 tweets were scrapped by Twitterscraper for each App, and each tweet contains information including User Name, User id, Number of Likes, Number of Replies, Number of Retweets, Tweet Text and Timestamp of Tweet.

3.2 Sentiment Analysis

Our first hypothesis was that the Google Play reviews with 4 stars and above would contain positive comments and reviews with 2 stars or below would contain negative comments.

In order to test this hypothesis, 3-star ratings were removed from the dataset. And Python library "Textblob" was used for processing textual data and uncovering the underlying sentiment with "polarity" score which ranges between -3(negative) and 3(positive), zero being the neutral sentiment. Next, a box plot of polarity scores was plotted by star ratings as shown in **Figure 1** below.

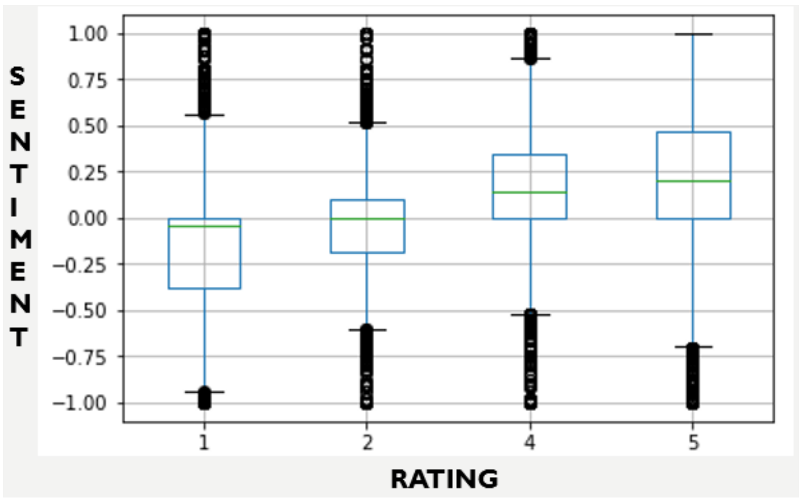


Figure 1: Box plot for Polarity score by different ratings

On observing the boxplot, it can be seen that as the star ratings increase, the polarity score has an upward trend from negative to positive. Hence, our hypothesis is proved right. In addition, it is noted that there are an appreciable number of positive sentiment(polarity) scores for 1-star rating and appreciable number of negative sentiment(polarity) scores for 5-star rating. This could be because the library is unable to detect sarcasm along with some potential data quality issues.

Our second hypothesis was that users would provide more details in negative reviews and only give short comments in positive reviews.

In order to test this hypothesis, the polarity scores calculated while testing the first hypothesis was used. In addition, the length of each review was stored in a separate column to judge whether it is a short review or a long one. As the length is a continuous variable, 5 buckets of the length were created as shown in following **Table 3.**

Text Length	Length Bin
0 – 10	Very Short
10 – 50	Moderately Short
50 – 200	Medium
200 – 700	Moderately Long
More than 700	Very Long

Table 3: Buckets of Text Length

Next, a box plot of Sentiment Polarity scores by length bin was plotted as shown in Figure 2 below.

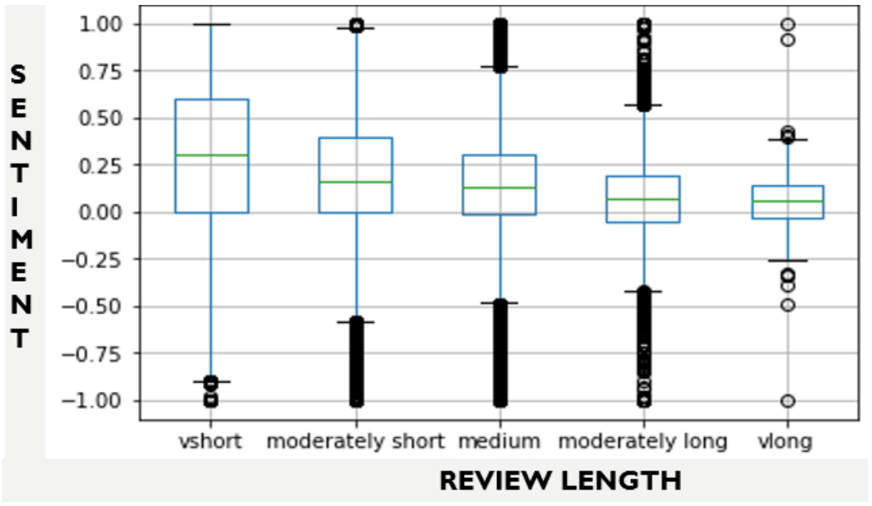


Figure 2: Box plot for Polarity score by different Review Length Bins

On observing the box plot, it can be seen that as the length of reviews increase, the polarity score has a downward trend from positive to negative. Hence, our second hypothesis is also proved right.

As to sarcasm, like any other type of natural language processing (NLP) analysis, context matters. Analyzing natural language data is the problem of the next 2-3 decades. It's an incredibly difficult issue, and sarcasm and other types of ironic languages are inherently problematic for machines to detect when looked at in isolation. It's imperative to have a sufficiently sophisticated and rigorous enough approach that relevant context can be taken into account. For example, that would require knowing that a particular user is generally sarcastic, ironic, or hyperbolic, or having a larger sample of the natural language data that provides clues to determine whether or not a phrase is ironic. Transparency in how the analysis occurs and additional background data such as the context of when data samples were gathered, what we know about the population that generated them, and so forth is important.

3.3 Trend of Star Rating

Our third hypothesis was the average star rating of an app would increase over time as the developers keep making improvements. If the developers could effectively receive feedbacks from user reviews and make updates and improvements to the app constantly, the user satisfaction level should be increasing over time. A plot summarizing the overall trend of yearly average ratings of all the apps is shown in **Appendix B**, it is noted that although the average rating for most of the apps have an upwards trend over years, there are still some apps experiencing fluctuations or even downwards trend in the average rating. This could be an indication that some of the developers have difficulties to effectively draw useful information from user reviews or understand users' expectations.

3.4 TF-IDF

To give a clearer view on where the developers could draw more useful insights of the user experience and receive feedback on how to improve the apps, our fourth hypothesis was tested, which assumed that Reviews between 2 and 4 stars ratings would provide more insightful details, as these users take more considerations before giving an outright bad or good rating (i.e. 1 or 5 star). To test on the hypothesis, the original dataset was separately into two group, 1 & 5 star reviews and 2 to 4 star reviews, it is noted that the average length of all the 1& 5 star review is 54.92 while the average length of all the 2 to 4 star reviews is 93.06 which is significantly longer than the former. Furthermore, the review text of the two datasets were converted into TF-IDF matrix using TfidfVectorizer with both 1-gram and 2-gram features. Top 100 weighted words/phrases were extracted from the TF-IDF matrix and were shown as the word cloud in **Figure 3.1** and **3.2** below.

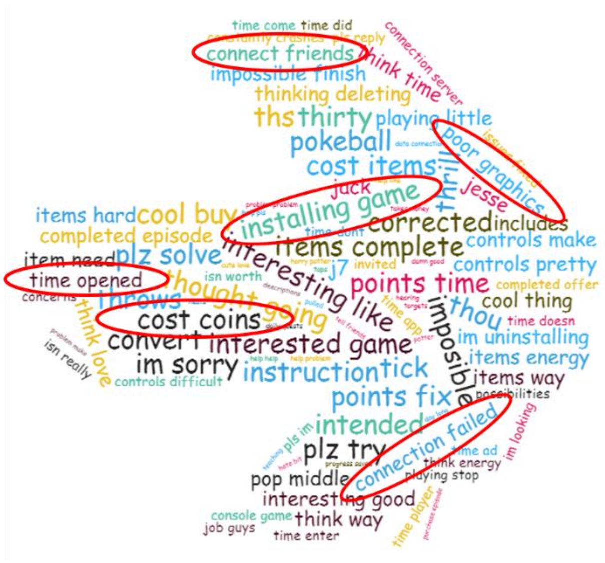


Figure 3.1: 2-4 star reviews

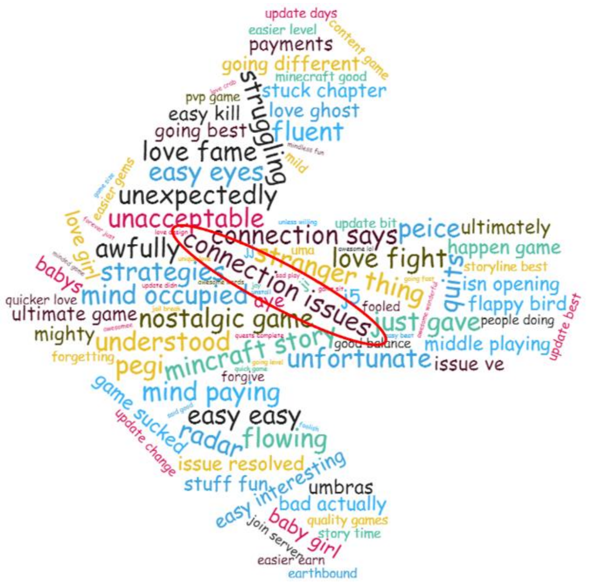


Figure 3.2: 1&5 star reviews

From the word clouds, it is found that the key words in 2-4 star reviews are generally related to the issues user encountered when playing the game, such as time opening the app might be too long, poor graphics, connection failed and issues with installing game. In addition, users may also mention why they like game such as it can connect friends. However, in 1&5 star review, although connection issues were also mentioned, most of the key words tend to be more subjective and emotional, such love, awfully, interesting and struggling, which could not provide too much useful information on how to improve the app. As such, our fourth hypothesis proved to be right, and developers should consider focus more on 2-4 star reviews to gain more insightful information from user feedbacks.

3.5 Random Forest and RNN Model

Based on the scored reviews from Google Play, a prediction function can be built to predict the scores of a review based on its text content automatically. This function could be useful when integrating reviews from other sources like Facebook and Twitter in which scores are not available.
Although the dependent variable in our dataset only contains 5 discrete values 1,2,3,4,5, which can be easily considered as a multi-class classification problem, a regression model is more suitable in this case because the 5 values are ordered numerical values. In addition, regression model can also predict a score more accurately instead of just giving a class, which can help to distinguish different reviews more easily.

To build the best regressor from the text content of reviews, traditional "bag of words" model with random forest algorithm and deep-learning based text RNN are experimented in our project.

3.5.1 "Bag of words" Random Forest Model

"Bag of words" random forest model is built with count vectorizer transformer and random forest regressor in scikit-learn package. The count vectorizer is set with min_df = 20 & only extracts English words, while Random forest is set as a default one. Naïve Bayes model is not used in this case because it can only be used in classification task.

3.5.2 Text RNN Model

Deep-learning based text RNN is also experimented with Keras in our case. This model is a simple LSTM model with time steps 40 and sequence returns, which means that all the reviews are padded into length 40 before they are put into the neural network model. Epoch is set to be 50 and batch size for training is 200. The structure of the text RNN Model is shown in **Figure 4** below.

Before the LSTM model, a fixed word embedding layer is built using the glove 200 dimension embedding dictionary. A trainable embedding layer is also tried, but it turns out that a fixed embedding layer brings better result since there are fewer parameters to train and model complexity is lower. Within the LSTM model, we set the number of hidden units to equal 128 and dropout ratio as well as recurrent dropout ratio to be 0.2 in order to avoid overfitting problems. The final dense layer takes relu as activation function to provide good regression results.

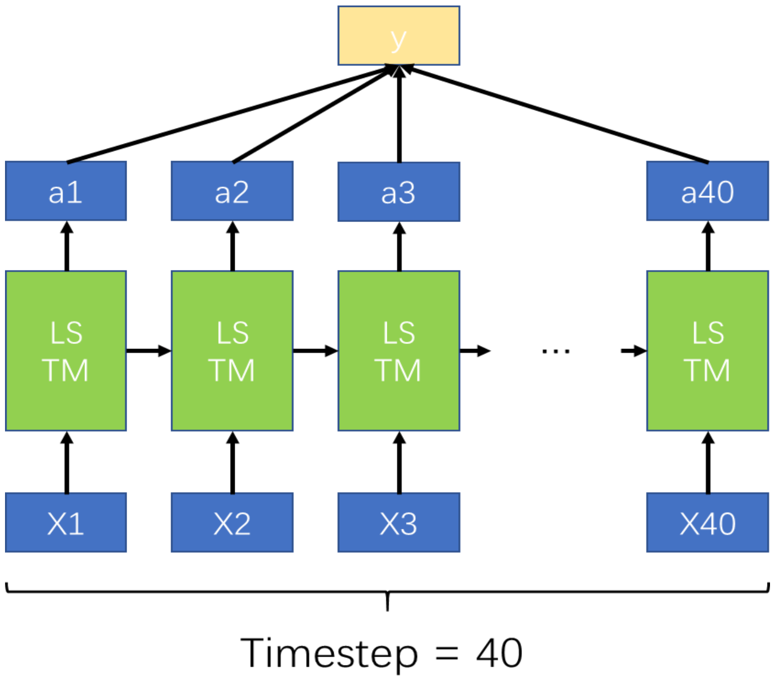


Figure 4: Structure of the RNN Model

3.5.3 Prediction results

In order to verify the validity of the two models, a train-test split with two thirds of training data and one third of testing data is implemented. The prediction result on the testing dataset is shown in **Table 4** below.

Model	RMSE	MAPE
Bag of words	0.95	25.85%
RNN	0.77	24.52%

Table 4: Prediction results of two models

As **Table 4** shows, text RNN performs slightly better than the bag of words model in this dataset. The RMSE of RNN method is 0.77, which can be interpreted as prediction result on average deviates from the true value by 0.77 score, while the prediction result of bag of words random forest model deviates from the true value by 0.93 score.

The MAPE (mean absolute percentage error) can be interpreted as the incorrect prediction percentage, which also shows that RNN performs 1% better than bag of words model.

3.6 Topic Modelling of Tweets

The process of topic modelling attempts to capture a list of many topics, i.e. statistical word clusters, that would describe a given set of texts. For our dataset of tweets, the most popular algorithm for topic modelling, LDA, has been used because it extracts key topics and themes from a large corpus of text. Here each topic is an ordered list of representative words, and the order is based on the importance of a word to the topic. LDA describes each document in the corpus based on the allocation to the extracted topics.
Here, the LDA model was trained for 5 topics, which was chosen through cross validation. Since this is an unsupervised learning application that creates the topics, human interpretation is required to evaluate if the topics makes sense. Following are the two topics reported that made most sense after the training.

Recently Updated Apps
New
Just
Time
Today
Update
Great
Latest

Gameplay
Gameplay
Gamedev
Mode
Playing
Detail
Fun
Game

Table 5: Topics selected from Topic Modelling results

These topics suggest that apps that have been updated recently by the developer, tends to receive reviews or tweets containing words like new, time, update, latest, great, suggesting that users might have been waiting for this update, and think it was great. Also, the other topic that contain top words like fun, mode, detail, can highlight certain keywords that can provide information to the developer on what features of the game have been well received, and which ones made them stop playing the game and uninstall the app.

4. Conclusion & Further Work

Based on the above analysis, it can be concluded that users would provide more details in negative reviews and only give short comments in positive reviews and the ratings given by the user are generally consistent with the sentiment involved in the text comments.

It is suggested that App developers should focus more on the 2-4 star reviews as it generally contains more detailed information on what exactly users like or dislike about the game. It is noted that there are a number of apps in our dataset having decreasing average ratings over years, which indicates some of the app developers could not effectively take feedback from users and fix the existing problems in time. Therefore, to be more efficiently receiving feedback from users, the App developers could provide certain key words which extracted from historical App reviews, such as connection issue, graphic issue, installation issue and app crash as some possible options to allow users simply select on the issues they encounter instead of typing a long comment. Different text vectorizer and settings of stop words could be tried to find out the best tools and parameters to identify key words from user reviews.

In this case of review score prediction, the deep-learning based text RNN method performs slightly better than the bag of words model. Thus, text RNN will be used as a regressor for new unscored reviews prediction from other sources. In fact, the potential of text RNN algorithm is still not completely discovered in this case. At the end of the model training process, validation loss is still on a slowly descending trend, but due to the hardware and training time limitation, it has to be stopped. In addition, reviews are only padded to 40 in order to avoid gradient vanishing problem when time steps are set to be too high, which will definitely lead to information loss. A better solution might be to use variable-length RNN or use CNN algorithm to perform this task.

In addition, the topics inferred during the topic modelling stage can be used to analyze the sentiment of their corresponding reviews to find out what the users are saying or feeling about the particular game updates/features. LDA model can also be used to create groups of users to group of features and can be used by the developers to created targeted marketing campaigns/advertisements to sell other applications with similar features.

.

References

Sangani Chirag, Ananthanarayanan Sundaram, "(2013) Sentiment Analysis of App Store Reviews", [online] Available: http://cs229.stanford.edu/proj2013/CS229-ProjectReport-ChiragSangani-SentimentAnalysisOfAppStoreReviews.pdf.

P. M. Vu, T. T. Nguyen, H. V. Pham, T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach", *Proc. 30th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 749-759, 2015.

Liu, J., Sarkar, M. K., & Chakraborty, G. (2013). Feature-based Sentiment analysis on android app reviews using SAS® text miner and SAS® sentiment analysis studio. In Proceedings of the SAS Global Forum 2013 Conference (Vol. 250).

Sefferman, Ashley (2016). "The Average Mobile Apps' Ratings and Reviews, by Category", [online] Available: https://www.apptentive.com/blog/2016/10/20/average-mobile-apps-ratings-and-reviews-by-category/

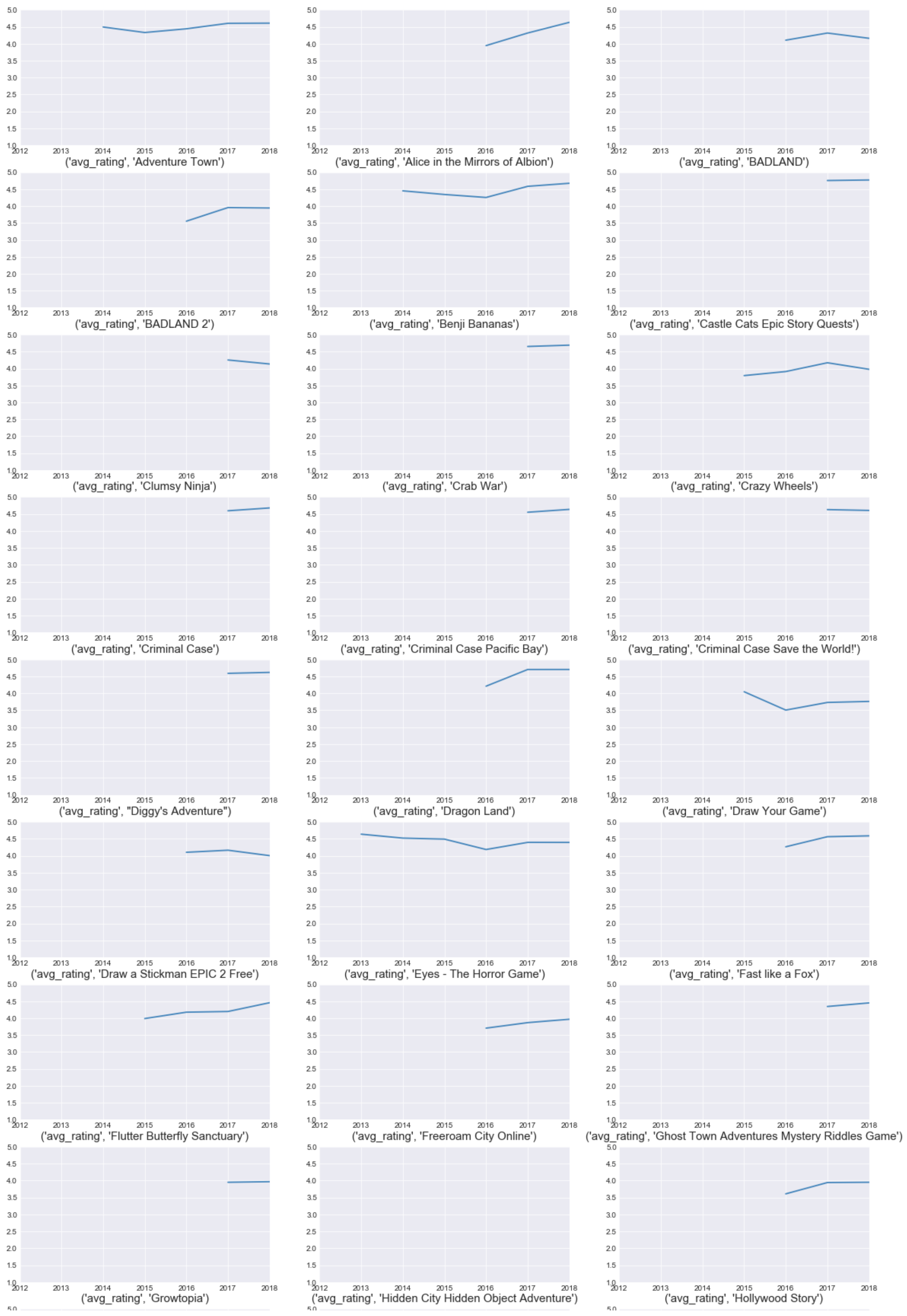
Appendix A – List of Apps in the Final Dataset

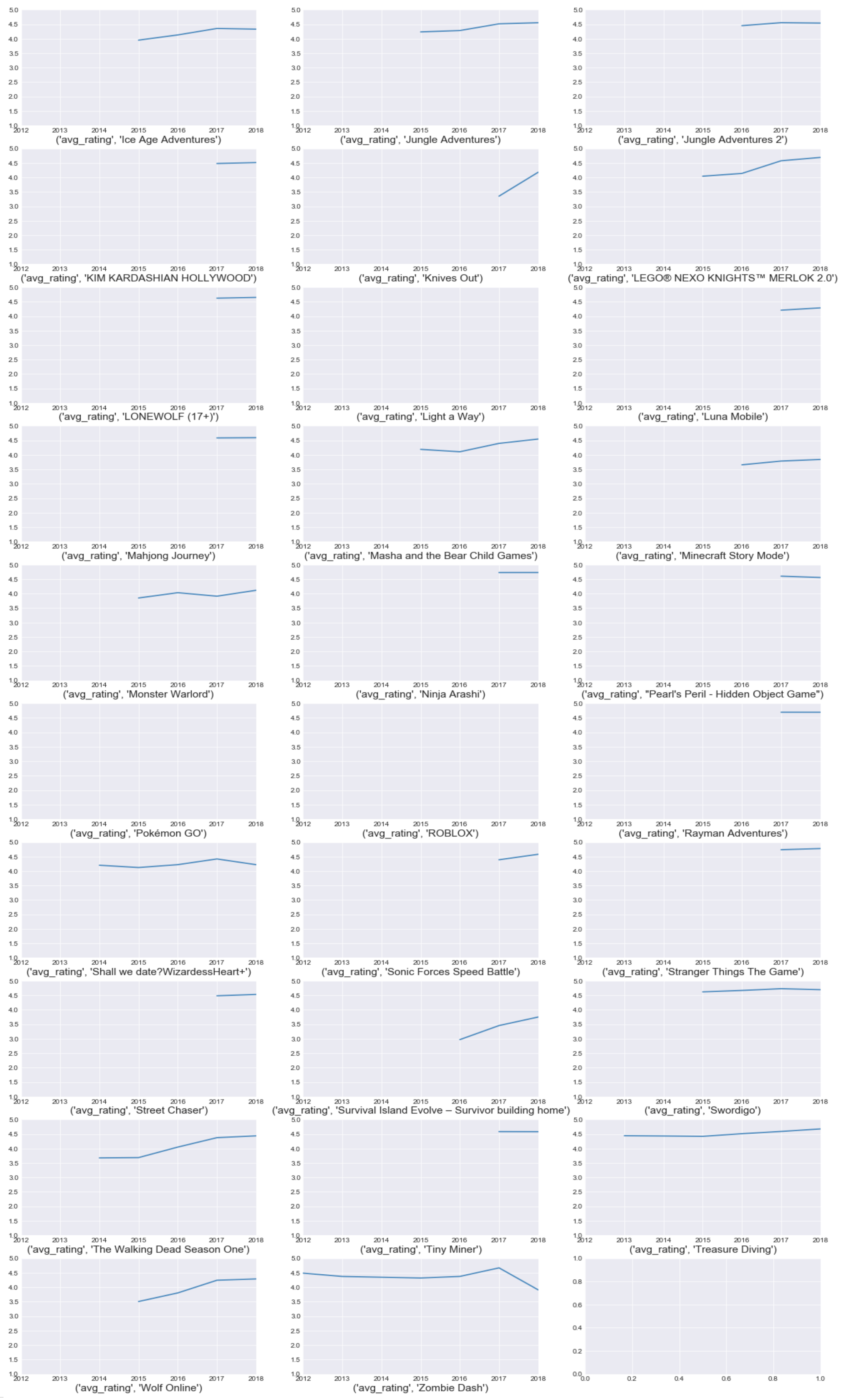
Number	App Name
1	Adventure Town
2	Alice in the Mirrors of Albion
3	BADLAND
4	BADLAND 2
5	Benji Bananas
6	Castle Cats Epic Story Quests
7	Clumsy Ninja
8	Crab War
9	Crazy Wheels
10	Criminal Case
11	Criminal Case Pacific Bay
12	Criminal Case Save the World!
13	Diggy's Adventure
14	Dragon Land
15	Draw Your Game
16	Draw a Stickman EPIC 2 Free
17	Eyes - The Horror Game
18	Fast like a Fox
19	Flutter Butterfly Sanctuary
20	Freeroam City Online
21	Ghost Town Adventures Mystery Riddles Game
22	Growtopia
23	Hidden City Hidden Object Adventure
24	Hollywood Story
25	Ice Age Adventures
26	Jungle Adventures
27	Jungle Adventures 2
28	KIM KARDASHIAN HOLLYWOOD
29	Knives Out
30	LEGO® NEXO KNIGHTS™ MERLOK 2.0
31	LONEWOLF (17+)
32	Light a Way
33	Luna Mobile
34	Mahjong Journey
35	Masha and the Bear Child Games
36	Minecraft Story Mode
37	Monster Warlord

38	Ninja Arashi
39	Pearl's Peril - Hidden Object Game
40	Pokémon GO
41	ROBLOX
42	Rayman Adventures
43	Shall we date?WizardessHeart+
44	Sonic Forces Speed Battle
45	Stranger Things The Game
46	Street Chaser
47	Survival Island Evolve – Survivor building home
48	Swordigo
49	The Walking Dead Season One
50	Tiny Miner
51	Treasure Diving
52	Wolf Online
53	Zombie Dash

Appendix B – Trend of Average ratings of Apps

Trend of yearly average rating




('avg_rating', 'Ice Age Adventures')
('avg_rating', 'Jungle Adventures')
('avg_rating', 'Jungle Adventures 2')
('avg_rating', 'KIM KARDASHIAN HOLLYWOOD')
('avg_rating', 'Knives Out')
('avg_rating', 'LEGO® NEXO KNIGHTS™ MERLOK 2.0')
('avg_rating', 'LONEWOLF (17+)')
('avg_rating', 'Light a Way')
('avg_rating', 'Luna Mobile')
('avg_rating', 'Mahjong Journey')
('avg_rating', 'Masha and the Bear Child Games')
('avg_rating', 'Minecraft Story Mode')
('avg_rating', 'Monster Warlord')
('avg_rating', 'Ninja Arashi')
('avg_rating', "Pearl's Peril - Hidden Object Game")
('avg_rating', 'Pokémon GO')
('avg_rating', 'ROBLOX')
('avg_rating', 'Rayman Adventures')
('avg_rating', 'Shall we date?WizardessHeart+')
('avg_rating', 'Sonic Forces Speed Battle')
('avg_rating', 'Stranger Things The Game')
('avg_rating', 'Street Chaser')
('avg_rating', 'Survival Island Evolve – Survivor building home')
('avg_rating', 'Swordigo')
('avg_rating', 'The Walking Dead Season One')
('avg_rating', 'Tiny Miner')
('avg_rating', 'Treasure Diving')
('avg_rating', 'Wolf Online')
('avg_rating', 'Zombie Dash')