**INTRODUCTION TO DATA SCIENCE**

**Applications reviews analysiss**

Vardit Arkash – 305134140

[vardit.arkash@mail.huji.ac.il](mailto:vardit.arkash@mail.huji.ac.il)

Gal Turgeman – 308222595  
[gal.turgeman@mail.huji.ac.il](mailto:gal.turgeman@mail.huji.ac.il)

Yarden Yagil - 311549083  
[yarden.yagil@mail.huji.ac.il](mailto:yarden.yagil@mail.huji.ac.il)

**Introduction:**

Our goal in this project is to find the main problems in an application, in order to enable the developers to improve their application in an easier way.  
We were focused on games applications from Google Play website. In order to implement it we have collected reviews from the applications, and by using NLP tools we tried to deduce what are the problems in the application, if there are any.

We defined 6 main problems to look for in the applications:

* Ads: there are to many advertisements in the application.
* Cost: users think the app doesn’t worth the money it cost.
* UX: bad design and inconvenient user interface.
* Appeal: the app is boring.
* Bug: there are bugs in the application.

We analyzed the reviews according to these 6 problems.

**The Data**

We first needed to extract reviews from games applications. For this purpose, we crawled the Google Play website (<https://play.google.com/store/apps>) and collected users reviews from various categories of games. The data we created consists of 200 applications and about 100 reviews for each one. The data size is X GB.

First the crawler goes through the html pages of various categories of games (i.e. action, board, cards, adventure, sport) and collects the urls of the applications. Afterwards it parses the html pages of the applications and collects the user's reviews of each application. Finally, all the reviews are saved in a dictionary which is the database that will serve us through out the project. The dictionary is structured so that the key is the application name and the value is a list of all the reviews of that application. The data collection is done once and the dictionary is saved in a file called "reviews.txt" which is located in the "files" folder.

**The solution description:**

In order to deduce what are the problems in an application, we evaluate negative feedbacks of the app by using 2-parts algorithm and some manual processing. The description will refer to each of the parts.

**Stage 1: Preprocessing the data.**

Our first task was to extract the data by crawling the Google Play webpage and extracting some useful information from it. We saved this data in two files:

1. reviews.txt – contains a dictionary with applications names as keys and list of reviews as values.
2. info.txt – contains two lists. The first one contains the apps' names and the second one the popular tokens that appears in all the apps' reviews.

In the code -

We have a preprocessing module that is responsible for creating the mentioned files. It is done by running two other modules:

1. crawler.py – crawling the website in order to extract the data.
2. topic\_extractor.py – given the data, it finds the popular tokens in it.

*All the code files for this level can be found in the 'preprocessing\_step' folder and the files that were created can be found in the 'files' folder.*

**Stage 2: Manual Processing.**

The work at this stage was done manually.   
  
We received the popular tokens list from the applications reviews from the previous stage, removed stop words and punctuations from it, and kept only words that are nouns or adjectives.

From this list we tried to figure out which words are the most useful for extracting info about the application's quality. We had words such as: "game", "play" and "coins" that couldn't help us figure out nothing about what the player liked or didn't liked in the game. In contrary, we had words like "time", "ads" and "money" that were more informative (remark num. 2).

We chose six main topics and for each topic we created a list of relative words and saved them in a new file:

* info\_manual.txt – contains two lists. The first one contains applications names (untouched by us) and the second one contains a dictionary that we created manually, where each topic name is a key and its value is a list that of words that are related to the topic.

*The new file can be found in the 'files' folder.*

**Stage 3: Data analysis.**

At this stage the user can choose one application from a given list. For the chosen application, we go over all of the application reviews and extract all the informative trigrams from it. For each trigram, we decide what its sentiment and if it's negative – we save it in a list.

At the next step, we go over the list of the negative trigrams and check for each topic (from the list we created in the previous level) how many trigrams are referring to this topic and creating a graph based on that info.

In the code –

We have a manager module that is responsible for creating the mentioned graph for the chosen application. It is done by running other modules:

1. reviews\_extraction.py – extracts the reviews for the application from the file that was created in the second stage.
2. Info\_extraction.py – extracts the informative phrases as trigrams.
3. sentiment\_analyzer.py – analyzes a phrase (trigram in our case) and determine if it’s a negative trigram.
4. result\_analyzer.py – plot the analysis results in graph for the given app.

*All the code files for this level can be found in the 'analyzing\_step' folder, and the graphs can be found in the 'Graphs' folder.*

Important remarks:

* The popular tokens were taken from all of the applications reviews and not per application.
* In the output of the first level, we received a list of words. In this list, we mentioned that some of the words were informative; they couldn't help us determine which is the main problematic topic.
* We mostly used python nltk package:

1. To extract the useful words from the reviews.
2. To determine the trigram's sentiments.

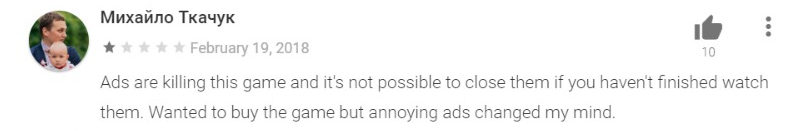
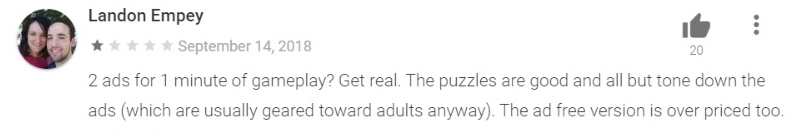
**Experiment**

* Evaluation Criteria: We measured the success of the project by a survey. The participants read the reviews of the applications and scored according to the reviews the problematic topics (The topics which selected in the first stage of the program). Afterwards we compared the participants decisions with the graphs obtained by running our program.
* Setup: We chose 6 applications from Google Play website that have different problematic topic. For each application, 3 participants were instructed to read all of its reviews and to select the topics that were problematic in the application from the following topics: Cost, UX, Ads, Appeal, Bug. The participants' had to score each topic between 0 to 10.
* Results: We noticed that the results were similar in both the manual experiment and in our program. The participants referred to the review's context and tone while the program analyzes the text by its words' sentiments. Overall, we could determine that the algorithm has achieved great results. In the visualization section, we will present some of the algorithm's outputs and later we will discuss its impediments.
* Visualization: In the next pages, we show a few examples for the algorithm results compared to the manual results we collected. In every example, the left figure represents the algorithm output while the right figure represents the manual results.

For the algorithm output we calculated how many negative trigrams referred to this topic and divided it by the sum of all the negative trigrams. For the manual results, we calculated the average between the participants rating and divided it by the sum of all the averaged ratings for each topic.

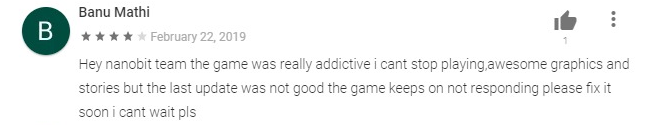
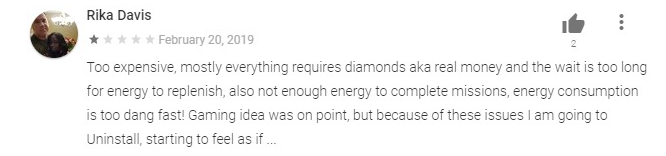
1. Baby puzzle:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Baby_puzzles_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Baby_puzzles_manual_plot.png |



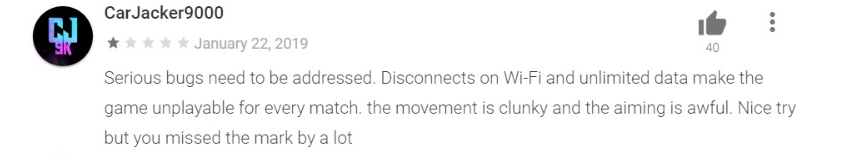
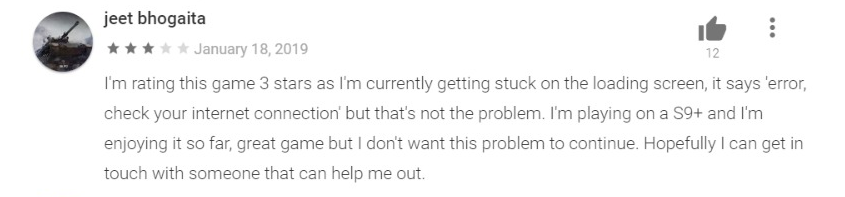
1. Hollywood story:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Hollywood_Story_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Hollywood_Story_manual_plot.png |



1. Standoff 2:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Standoff_2_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Standoff_2_manual_plot.png |



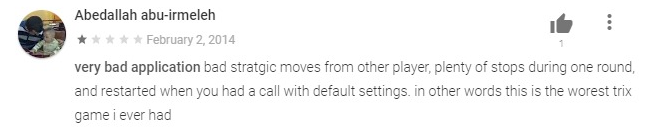
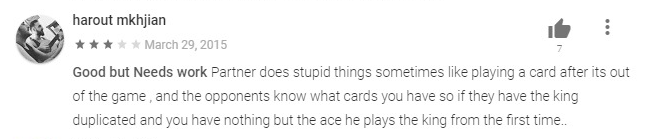
1. Street racing 3d:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Street_Racing_3D_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Street_Racing_3D_manual_plot.png |



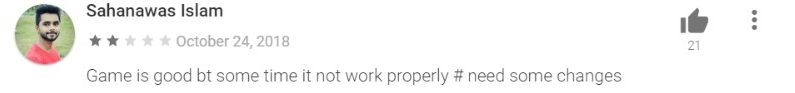
1. Trix:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Trix_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Trix_manual_plot.png |



1. Street chaser:

|  |  |
| --- | --- |
|  |  |
| C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\Street_Chaser_plot.png | C:\Users\Gal Turgeman\Desktop\Gal\שנה ד'\מחט בערימת דאטה\project\ReviewAnalyzer\Graphs\CompareResults\Street_Chaser_manual_plot.png |



* Impediments:

1. Some of the reviews were specific about the application's details. Adding this words would cause overfitting and may not fit to another application. The problem that the algorithm wouldn't catch these words as negative but the human participant would recognize it as issue that needs to be fixed. For example: In the 'Standoff 2' game, some reviewers asked to add more guns' models to the game. The algorithm ignored it but the participants marked it as UX issue (as you can see in figure 3).
2. The applications we chose in this part were top rated, thus most of their reviews were positive. The graphs represent the negative reviews' distribution by the topics above. That may be the reason why some of the algorithm's columns higher than the participants'; the participants saw these reviews as insignificant.
3. The participants could capture a slang or double meaning in the reviewer's words while the algorithm treats them as written, without a hidden meaning, or simply ignore them.