Will It Sell – Mobile Application Success Prediction System

Shreyam Kela
Department of Computer Engineering
San Jose State University
San Jose, USA
shreyam.kela@sjsu.edu

Sayalee Bhusari
Department of Computer Engineering
San Jose State University
San Jose, USA
sayaleeshankar.bhusari@sjsu.edu

Vaishali Koul
Department of Computer Engineering
San Jose State University
San Jose, USA
vaishali.koul@sjsu.edu

Harsh Agrawal
Department of Computer Engineering
San Jose State University
San Jose, USA
harsh.agrawal@sjsu.edu

***Abstract*—With an ever-increasing dependency on mobile phones; mobile applications *a.k.a* mobile apps have become an inherent part of our daily lives. Consequently, developing and releasing innovative and marketable apps, requires a large amount of investment on both the development, and the market research as well. Due to a lack of an appreciable amount of investment on the latter, brilliant app ideas fail to attain any significant amount of profit, when deployed on the app store. This paper describes a system that performs a detailed market trend research and provides predictions for mobile application developers, by analyzing the idea or the description of the app that they propose to develop. The system helps the developers in assessing the selling ability or the performance of their app idea, practically before developing and deploying their app on the app store.**

Index Terms—Mobile Application, App, App Store, App Idea, Prediction, Selling Ability, Text Mining, Text Analytics, Similarity, Cosine Similarity, Tfidf, Nearest Neighbour, NLP

I. Introduction

Prediction systems are rooted on logic based deductions. In the ideal case, all the apps that are very similar should perform equally on the app store. This is the fundamental idea behind our prediction model. As we include real world constraints such as age-group focus area, geographical focus area, app size, and so on, into the model, it gets closer to the actual market performance.

Apple app store provides app analytics information but it is only for the apps that are already on the app store. Our prediction system goes one step ahead and analyses the performance of the past apps that are similar to the new idea, and provides a detailed analysis of the potential selling ability of the new app on the app store.

II. Architecture Flow

Users interact with our model through a web portal named 'Will It Sell', which has the following architecture:

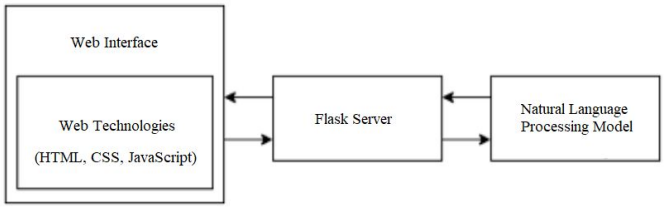

Fig. 1. Architecture Flow

III. Technology Components

The technologies that we have used to develop the web portal for our system are: Frontend - HTML, CSS, and JavaScript with ReactJS; Backend - Flask as Server, and Natural Language Processing model to generate predictions.

The Natural Language Processing model is the most important part of our prediction system. The fundamental task of the model is to find descriptions in the training dataset, that are similar to the newly entered app idea or description.

A threshold value decides the number of similar apps that our model will finally analyse to provide predictions. The threshold value depends on a number of constraints such as whether the similar apps are free or paid, whether the apps are of a specific genre, whether the apps possess a specific functionality, and so on. The weighted average of the features of the similar apps, is the final prediction of our model. The weights are defined by the similarity percentage of the similar apps, and the real world constraints specific to the app genre.

IV. Model Components

The following block diagram shows the major components of our prediction model:

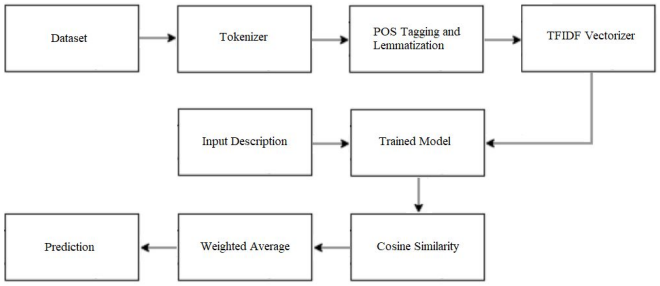

Fig. 2. Prediction Model

A. Dataset

The first component of the model is the dataset. Any prediction model is as good as the dataset being fed into it. The dataset that we have used for our study is a web crawled dataset of 7196 mobile apps from the Apple App store, which is a good amount to provide decent predictions. The dataset set includes the full name of the mobile app, the app description, total number of installs, app rating, number of ratings, age-group focus area, price, version, genre, reviews, app size, and a few more features.

B. Tokenizer

We use the *RegexpTokenizer* of the Natural Language Toolkit Library, to clean up the stop words, punctuations, and special characters, in all the app descriptions of the dataset and tokenzise each word to form the vocabulary of our model. This is a preprocessing step.

C. POS Tagging and Lemmatization

Before forming our vocabulary, there is another preprocessing step. We feed each tokenized word into a Part-of-speech tagger and then lemmatize it to reduce the word to its lemma. This decreases the size of the vocabulary significantly, which is necessary in order to obtain a fast performing model. This largely reduces the time taken for each prediction at the trade off of a negligible reduction in prediction accuracy.

D. TFIDF Vectorizer

Each app description in the dataset is fitted to form the vocabulary. After the vocabulary has been formed, each description is reduced to a unique TFIDF vector with the help of the word frequencies of the different words in each description, and the word frequencies of the same words in the whole dataset. For this we use the *TfidfVectorizer* of the scikit-learn machine learning library. We include a max_features constraint value in the *TfidfVectorizer* to reduce the vocabulary for faster processing. It is found out that if the max_features is kept at a value corresponding to the under-root of the total words in the vocabulary, the prediction accuracy does not decrease.

E. Trained Model

The multi-dimensional array containing the tfidf vectors for each of the descriptions form the basis of our trained model. With the help of 80% training set and 20% testing set, we perform the training and testing of our model.

F. Input Description

The new description of an app that requires predictions, is input into our model. The model first preprocesses it to reduce it to a TFIDF Vector that will be compared with the documents or the vectors already present in the trained model

G. Cosine Similarity

Using the cosine of the angles between the new vector and all of the vectors in the model, we determine the similar documents, according to the threshold factor that decides the number of similar documents to output.

H. Weighted Average

A weighted average for each of the to-be-predicted feature is found out for each of the similar app description. Similarity percentages of the similar apps decide the weights, in accordance to the real world constraints specific to the app genre, present in the dataset.

I. Prediction

The weighted averages of the features are the final predictions, that are sent to the frontend.

The predictions are sent in the form of a JSON file, by the Flask Server. At the frontend, ReactJS parses the JSON input and forms the different interactive insights and graphical analytics and displays it on the web interface with the help of HTML and CSS.

V. Prediction Insights And Web Interface

1. Main Page:

The main page of the web interface of ‘Will It Sell’. A text box is provided to enter the app description

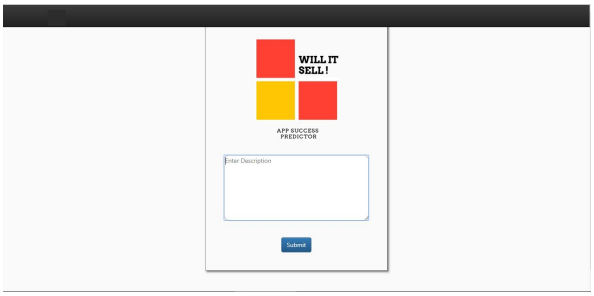

Fig. 3. Main Page

2. Backend Prediction Log:

The backend prediction result log is shown here. The queried application is an Airport Assistant app. Approximately 10 lines of the app description was submitted onto the main page of 'Will It Sell', to test the model.

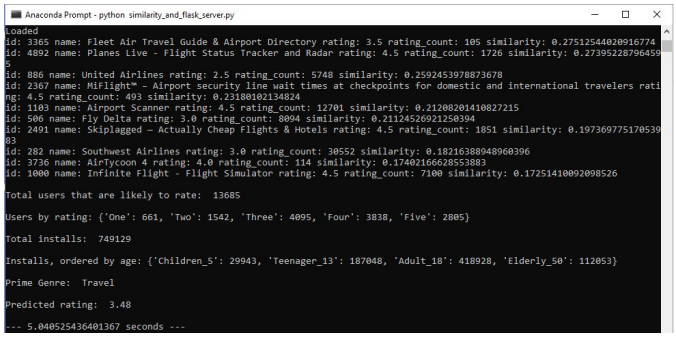

Fig. 4. Prediction Log

3. Predictions Overview Page:

The Predictions tab show an overview of the predictions. It shows the potential rating of the new app, the number of downloads, the genre, the number of users that might rate the app, the top 3 similar apps, and the overall selling ability.

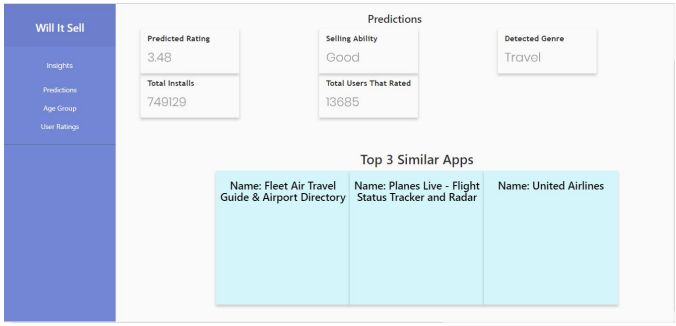

Fig. 5. Predictions Overview

4. Similar App Insights:

The predictions tab also shows insights about the top 3 similar apps already present on the app store, when we hover on their cards. The cards show the rating of the similar app, their similarity score, and a section from their app descriptions.

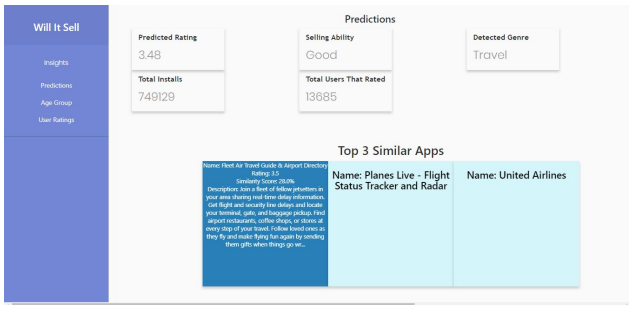

Fig. 6. Similar App Insights

5. Graph of Installs by Age Group:

The Age Group tab shows the prediction graph of total number of installs by age group.

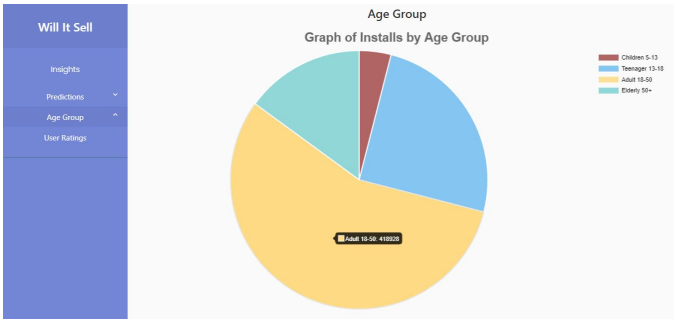

Fig. 7. Graph of Installs by Age Group

6. Graph of Number of Users by Rating:

The User Ratings tab shows the prediction graph of total users by ratings.

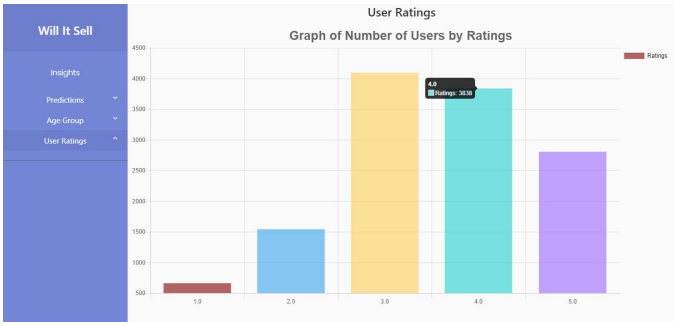

Fig. 8. Graph of Number of Users by Rating

The web interface of the Prediction System provides a detailed amount of information about the potential selling ability of the new app idea, that the developer can use to adjust and adapt their app idea to make the app more successful.

VI. Future Work

Professor Rakesh Ranjan has helped us list out a few ideas for inclusion into our system to increase the prediction accuracy and enhance the functionality of our system:

1. Include more real-world constraints into our model, such as the brand value of the firm that has developed the app.
2. Use Word Embeddings in the Text Mining process to increase the prediction accuracy
3. Display insights on more similar apps, as a premium feature.

VII. Conclusions

The project has helped us understand how Natural Language Processing models are created and how they are integrated into full fledged web interfaces. It has also instilled design thinking into us, which is necessary to market our system as a sellable product.

VII. Acknowledgements

We would like to thank Professor Rakesh Ranjan for encouraging us into developing such an innovative application and helping us in all of the phases of its development.

References

[1] Text Analytics - https://blog.exploratory.io/demystifying-text-analytics-preparing-document-and-term-data-for-text-mining-in-r-4f858feb4b77#.45wasxkwt

[2] Tfidf Tokenizer - https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] Statistical NLP - http://billchambers.me/tutorials/2014/12/22/cosine-similarity-explained-in-python.html