Analysis of Google Play store review

Sanket Prabhu: srp140430
Chirag Chudasama: cbc140130
Kunal Kapoor: kxk140330

Abstract:

Developing an app in today's smartphone ecosystems is easier than ever - the barrier of entry is low enough that a single developer can write a commercially successful app. The iOS App Store is reported to have reached the one million apps mark. As of October 2013, 50B apps have been downloaded from the app store. Furthermore, revenue earned by developers from the iOS App Store have been over $1.5B in 2012. Customer interest in apps is evident: top apps in the Google Play app store have over 1m reviews.
As a developer, it is essential to "stay on top of your game", i.e., keep your app updated with the most requested features and bug-fixes. However, most app stores provide only an average rating (out of 5) for each app. Consequently, it is difficult to identify why people like or dislike a particular. We aim to solve this problem.

Problem Statement:

The aim of this project is to collect reviews from Google Play Store and preprocess it and analyzing it to give top 30 positive as well as top 30 negative reviews about particular app, and to provide more detailed analyses of the reviews of an app to a developer, beyond an average rating.

System Overview:

1. To get review about particular app.

We are planning to scrape reviews for popular apps off the Google Play store using App ID.
We are planning to use Java/Python to retrieve reviews.

2. Processing reviews.

A review consists of the following relevant data:

a. A unique author ID,
b. Review Creation Time
c. Rating (ranging from 1 to 5)
d. Review Text

We normalize the ratings to a scale of [0, 1].
Expectedly, not all reviews possess perfect grammar or punctuation, rendering preprocessing a must. To begin, text for each review is transformed to lowercase. Subsequently, emoticons are converted into symbols. Emoticons can potentially prove to be strong indicators of opinion, and thus, are valuable. Now, exclamation marks are transformed into symbols, since we believe that they, too, can prove to be strong indicators of opinion. Finally, the review text is stripped of punctuation, and replaced by whitespace.

3. Creating a Corpus of Subjective Words

We hypothesize that the opinion of a review is indicated by the presence of "subjective words". For example, a review is more likely to complain about crashes, and have a low rating, if it contains the word "crash". Thus, we intend to create a corpus of such "subjective" words.
We also hypothesize that a subjective word is indicative of a high or low rating.
For example, a review with the word "crash" in it is likely to have a low rating, whereas one with "stable" in it is likely to have a high rating.
Consequently, we will extract subjective words from reviews by analyzing their power to predict the rating of a review. Specifically, we are planning to apply logistic regression to review ratings.

4. Identifying Topics in a Review

Given a review text, a list of subjective words, and a list of topics that each subjective word corresponds to, the topics addressed by the review are the union of topics for each word in the review.

5. Ranking Topics in Order of Relevance

Given a list of topics, and a list of reviews associated with each topic, one can compute various metrics for each topic, such as average rating, number of reviews, etc. Consequently, one can rank the topics in order of relevance to a single or hybrid metric. For example, a hybrid metric that ranks topics in increasing order of average rating and decreasing order of number of reviews provides a list of topics that customers are most unhappy about. Similarly, hybrid metrics can be designed to provide topics that customers are most happy about, or write the longest reviews about, etc.

6. Identifying Representative Reviews for a Topic.

For each topic, we are planning to identify a list of representative reviews that summarize public sentiment on that topic. This is achieved by ranking all reviews relevant to a topic according to some metric and picking the top 3. We will chose a metric that ranks reviews in proportion to the number of times a word from the topic set appears in the review, and inversely proportional to the squared distance of the review's rating from the average rating for that topic.

Predicted Output:

We started out with the goal of providing a more detailed analyses of the reviews of an app to a developer, beyond an average rating.
We will be able to provide a list of topics that are relevant to the manner in which the developer wishes to interpret the reviews, an average rating that conveys general sentiment for that topic, and representative reviews that give insightful criticism regarding the topic.

Software/Tools?

- Python IDE
- Java (JDK)

References:

[1] C. Jones, "Apple's App Store About To Hit 1 Million Apps," 11 12 2013. [Online]. Available: http://www.forbes.com/sites/chuckjones/2013/12/11/apples-app-store-about-to-hit-1-million-apps/. [Accessed 12 12 2013].

[2] R. Baldwin, "Apple Hits 50 Billion Apps Served," 15 05 2013. [Online]. Available: http://www.wired.com/gadgetlab/2013/05/apple-hits-50-billion-served/. [Accessed 12 12 2013].

[3] Canalys, "11% quarterly growth in downloads for leading app stores," 08 04 2013. [Online]. Available: http://canalys.com/newsroom/11-quarterly-growth-downloads-leading-app-stores. [Accessed 12 12 2013].

[4] L. Zhuang, F. Jing and X.-Y. Zhu, "Movie review mining and summarization," in Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06), ACM, New York, NY, USA, 2006.

[5] H. Tang, S. Tan and X. Cheng, "A survey on sentiment detection of reviews," Expert Systems with Applications, vol. 36, no. 7, pp. 10760-10773, 2009.

[6] "android-market-api," 04 11 2013. [Online]. Available: http://code.google.com/p/android-market-api/.