Sentiment Analysis of App Store Reviews

Chirag Sangani csangani@stanford.edu
Sundaram Ananthanarayanan sananth2@stanford.edu

Abstract – Analyzing user sentiments towards apps through their review comments and ratings can be economically profitable to app developers. We propose a system that provides a many-to-many mapping from reviews to topics of interest, and a list of reviews for each topic that are representative of user sentiment towards that topic.

1. Introduction

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 [1]. As of October 2013, 50B apps have been downloaded from the app store [2]. Furthermore, revenue earned by developers from the iOS App Store have been over $1.5B in 2012 [3]. 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.

Section 2 talks briefly about related work. Section 3 outlines our methodology in detail. In section 4, we analyze our methodology in light of experimental results. Finally, we conclude in section 5 with a mention of potential improvements to be explored in later work.

2. Related Work

A number of articles have been published on the sentiment analysis of movie [4] and product [5] reviews, with elaborate consideration towards natural language processing and understanding, subjectivity detection and opinion identification, feature selection and extraction, classification, language models, etc. [6] A subset of these articles focus on topic considerations. i.e., determining the topic of a document. This issue bears relevance to our system, which is essentially a ranked topic classifier. [6] enumerates a number of prior work on this topic using varied approaches in section 4.9.1.

3. Methodology

Our methodology is a multi-step process that addresses two tasks: the identification of topics, and extraction of opinions associated with these features. Consequently, our methodology is a multi-stage process consisting of the following steps:

(1) Creating a repository of reviews,
(2) creating a corpus of subjective words,
(3) identify topics of interest,
(4) identifying topics in a review,
(5) ranking topics in order of relevance, and
(6) identifying representative reviews for a topic.

3.1 Creating a Repository of Reviews

3.1.1 Downloading Reviews

To create a sufficiently large dataset, we scraped reviews for popular apps off the Google Play store. We were assisted by `android-market-api` [7], an unofficial Java library that allows for direct access to Google's official Android market servers for information.

`android-market-api` provides the following relevant services:

a. Querying for App IDs based on search queries, and
b. Downloading reviews for apps based on App IDs.

Due to Google's anti-spam, anti-DDOS policies, there are certain limitations on harvesting data:

a. Not all apps show up when querying for App IDs. For example, querying for "angry birds" does not return results for the popular "Angry Birds" game series.
b. Only a maximum of 4500 comments can be downloaded for any app.

We collected reviews for the following apps: Audible, Flixster, Jewels, Opera Mini, Pandora, Soundhound, and Yelp.

3.1.2 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.

3.2 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 extract subjective words from reviews by analyzing their power to predict the rating of a review. Specifically, we apply logistic regression to review ratings.

For each review x, the set of features $\boldsymbol{\phi}(x)$ is the set of all unigram features extracted from the review text. We used presence as a feature rather than frequency – that is, a feature value is 1 or 0 if the related word (or pair) is present or not. As an optimization, we omit common stop words, obtained from [8].

The hypothesis function is the sigmoid function. That is, given a feature vector $\boldsymbol{\phi}(x)$ and weight vector $\boldsymbol{\theta}$, our prediction $h_{\boldsymbol{\theta}}(x)$ of the normalized rating of x is:

$$h_{\boldsymbol{\theta}}\big(\boldsymbol{\phi}(x)\big) = g\big(\boldsymbol{\theta}.\boldsymbol{\phi}(x)\big) = \frac{1}{1+e^{-\boldsymbol{\theta}.\boldsymbol{\phi}(x)}}$$

The weight vector $\boldsymbol{\theta}$ is learnt using stochastic gradient descent:

$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} + \alpha\left(y^{(i)} - h\left(\boldsymbol{\phi}\left(x^{(i)}\right)\right)\right)\boldsymbol{\phi}\left(x^{(i)}\right)$$

Here, α is the learning rate.

The error for the set $T \in \{\text{Test, Training}\}$ is defined as:

$$E_{\boldsymbol{\theta},T} = \frac{1}{|T|}\sum_{j\in T}\left(y^{(j)} - h_{\boldsymbol{\theta}}\left(\boldsymbol{\phi}\left(x^{(j)}\right)\right)\right)^2$$

We perform stochastic gradient descent for multiple passes until:

$$\left|E_{\theta_{i+1},Test} - E_{\theta_i,Test}\right| < \epsilon$$

To summarize, there are three parameters to our algorithm: α, ϵ and $\boldsymbol{\phi}$.

Once convergence is reached, we dump the weight vector $\boldsymbol{\theta}$ to a file. This vector is a map on words to scalar values, and is interpreted as a list of words sorted on rating influence – the top few words indicate a positive rating, and the bottom few words indicate a negative rating.

3.3 Identifying Topics of Interest

From the previous segment, we have a list of subjective words that indicate a positive or negative opinion. In this segment, we will map these words to topics of interest. Given that we lack a database of semantic knowledge that maps a subjective word to a relevant topic, and our inability to use NLP techniques to infer topics from sentences (see 4.2.2), we decided to make use of WordNet [9].

WordNet allows us to group words into "synsets". Each synset is a group of synonymous words. We hypothesize that words that are synonyms represent the same topic. Hence, a topic is but a set of synonyms.

We use the following algorithm to compute topics / synonym sets:

$$\Gamma \coloneqq \bigcup_{w\in\boldsymbol{\Phi}} synsets(w)$$
$$\Delta \coloneqq \{s\colon \phi | s \in \Gamma\}$$
for all s in Γ:
  for all w in $\boldsymbol{\Phi}$:
    if s in $synsets(s)$:
      $\Delta(s) \coloneqq \Delta(s) \cup \{w\}$

$\nabla := \{\Delta(s) | s \in \Gamma\}$
return ∇

Here, $\mathbf{\Phi}$ is the set of all unigram features obtained from 3.2.

3.4 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.

3.5 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.

3.6 Identifying Representative Reviews for a Topic.

For each topic, we 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 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.

4. Analysis

4.1 Algorithmic Complexity

The first step in our pipeline is linear regression on unigram features. This step is trivially $O(nk)$ where n is the number of reviews, and k is the average length of each review, for a fixed learning rate.

In the next step, we compute a list of topics from word synsets. Assuming that it takes constant time to compute word synsets, the time complexity is $O(sk)$, where s is the number of synsets and k is the number of unigram features.

The remaining two steps are mere rankings, computed in linear / polynomial time. Hence, our pipeline is algorithmically efficient.

4.2 Output

4.2.1 Creating a Corpus of Subjective Words

The first step in our pipeline – linear regression – gives us the basic ability to predict the sentiment of a review. This is useful to identify features in a review that are good indicators of sentiment.

Our values for the parameters of linear regression were: $\alpha = 0.0001$, $\epsilon = 0.001$, $\boldsymbol{\phi}$ is the presence-indicating feature vector of all unigrams with stop words filtered out. Finally, our training set was 90% of all reviews in the repository, and the test set was the remaining 10%.

With these values, we reached convergence in approximately 20 iterations, and our test / training set error was ≈ 0.15.

The top 30 positive words were:

love, great, good, awesome, best, !, excellent, app, nice, game, browser, cool, fast, easy, works, ☺ (and variants), fun, amazing, movie, opera, addictive, perfect, music, movies, awsome, super, helpful, fantastic, better, jewels.

The top 30 negative words were:

update, work, open, sucks, will, phone, uninstall, ads, play, bad, songs, books, poor, crap, book, crashes, useless, screen, uninstalled, force, terrible, download, 3, load, horrible, uninstalling, start, takes, waste, annoying.

Subjectively, it is obvious that the top 30 positive words indeed convey a "positive sentiment", whereas the bottom 30 words convey a "negative sentiment".

Once we obtained the requisite features, we mapped them to topics. Previously, we included a step where features whose weights were ~ 0 were removed using k-means clustering, since these features had poor rating predictive power. However, this does not necessarily mean that these features are irrelevant – they could be indicative of mixed opinion, and are just as valuable. Additionally performing this step led to an insufficient number of topics left for the remaining steps.

Consequently, we decided to leave them in, and filter out stop words using a fixed dictionary.

4.2.2 Identifying Topics in a Review

Topic inference in a document is a well-studied task. However, most approaches assume a structured document with well-formed grammar and only the occasional error. However, most reviews we encountered showed poor structure, spelling and syntax. This rendered NLP techniques impossible. Consequently, we fell back to a simple technique – finding a group of subjective words with similar meanings in a review, and interpreting this set of words as a topic of discussion.

We considered mapping each such set of words to a hand-picked list of "super-topics" manually. However, this proved too arduous, and defeated the purpose of our efforts to automate everything. Hence, this technique was discarded.

Using our technique, we obtained an average of 2000 topics per app. Note that this list contains a large spectrum of all possible topics, ranging from "ads", "bug" or "crash" all the way to "fine", "okay", etc. In the next step, we rank topics according to their relevance.

In our results, we noticed that not all topic sets are obviously a set of synonyms. For example, the set ('bring', 'bringing', 'brings', 'brought', 'play', 'played', 'playing', 'plays', 'work', 'worked', 'working', 'works') does not obviously contain pairwise synonymous words. However, one must realize that a word can be used in many senses, and can imply different meanings. In the absence of contextual meaning of a word, one must assume every meaning to be possible, leading to a certain level of ambiguity. This might also lead to two or more independent topics fused together under the same topic. However, this issue is not prevalent or alarming: the example above is the worst-case scenario.

Another issue we noticed was some topics were similar for the most part, except for one or two words, and hence appeared together in rankings everywhere. However, this issue was not too prevalent either, and hence, was deferred.

4.2.3 Ranking Topics in Order of Relevance

The relevance of a topic depends on the desideratum of the developer. A developer may desire to look at topics with negative sentiments, positive sentiments, or topics that are popular, or any other metric.

We defined metrics for some such use-cases. For example, topics with negative sentiments can be obtained by ranking them proportional to the number of relevant reviews, and inversely proportional to the average rating. This metric is not entirely statistically sound – any topic that has enough reviews will have a mean close to the average rating – however, it works well in practice.

For Pandora, the top negative topics we found were: ('garbage', 'refuses'), ('pathetic', 'poor'), ('email', 'emailed', 'emails'), etc.

In practice, we found another heuristic extremely effective: topics ranked such that they are far from the extremes in terms of ratings, and are far from the app average, and have a sizeable number of reviews, provide an alternate view of the public sentiment that is fairly critical and insightful. For example, for Pandora, the top topics according to this metric were: ('play', 'played', 'playing', 'plays',…), ('ad', 'ads', 'advertisement', 'advertisements', 'advertising', 'adverts'), ('call', 'song', 'songs'), etc. We believe that this metric provides good results because:

(a) Most reviews that provide a 1 star / 5 star rating do so without providing any insightful criticism. On the other hand, reviews with 2-4 star ratings provide more detailed criticism. One can hypothesize that a 2-4 rating indicates the author put thought into the rating before choosing an outright good/bad (leading to a 5/1 star rating respectively) distinction, and this same thought is usually reflected in the review itself.

(b) If the average rating is high (as is the case with all the apps we happened to choose), then most reviews close to the app average, again, fail to criticize the app properly.

Other means of ranking topics could be conceived of, such as measuring the "insightfulness" of a review, and computing the average for a topic, however, such avenues were deferred. Additionally, we would like replace the use of a heuristic with that of a learning technique. However, in the face of lack of labeled data, we are unable to do so.

4.2.4 Identifying Representative Reviews for a Topic

To identify reviews that are most representative of a topic, we use the tf-idf (term frequency, inverse document frequency) metric for all terms in a topic, combined with a metric that ranks reviews in order of

their square distance from the topic's average rating. Note that for any topic, the document frequency for a term is constant for all reviews, so only the term frequency in a review is helpful.

Using this metric, we get fairly accurate results for the following topics for Pandora (in processed form):

('play', 'played', 'playing', 'plays',…):

absolutely love it but the widget never loads anymore and sometimes it will stop in the middle of a song and go to the next song without me doing anything or else it will play an ad while one song is playing and another song will start playing on top of that another problem is that the app played 3 different songs at one time 3x in 1 hour not sure whats going on

('ad', 'ads', 'advertisement', 'advertisements', 'advertising', 'adverts'):

indecent inappropriate ads! i like the application however there needs to be a way to opt out of your ads that contain sensual provocative advertisement for social related content because of the nature of your ads your application needs to be rated for adults only your ads are inappropriate for children and minors

Subjectively, one can conclude that these results are fairly relevant to the topic at hand. This is because the metric ensures that the representative review is one that talks most about the topic at hand. Finally, one can argue that the review is representative since it has a rating close to the topic's average rating.

5. Conclusion

To summarize our work, 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 are 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.

There are a number of avenues available for improving our methodology. We can improve the manner in which we infer topics from a review by applying NLP techniques on structured data, and the manner in which we signify relevance of topic or representativeness of a review using labeled data. We would like to acknowledge Dr. Andrew Ng (instructor, CS 229), Dr. Percy Liang (instructor, CS 221), the assistants for these courses, and our course mates, for their help, guidance, and feedback.

6. 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] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1-135, 2008.

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

[8] Ranks.nl, "English stopwords," [Online]. Available: http://www.ranks.nl/resources/stopwords.html. [Accessed 12 12 2013].

[9] Princeton University, "About WordNet," 2010. [Online]. Available: http://wordnet.princeton.edu.