Why People Hate Your App — Making Sense of User Feedback in a Mobile App Store

Bin Fu, Jialiu Lin, Lei Li[†], Christos Faloutsos, Jason Hong, Norman Sadeh
{binf, jialiul, christos, jasonh, sadeh}@cs.cmu.edu, [†]leili@cs.berkeley.edu
School of Computer Science, Carnegie Mellon University, [†]University of California, Berkeley

ABSTRACT

User review is a crucial component of open mobile app markets such as the Google Play Store. How do we automatically summarize millions of user reviews and make sense out of them? Unfortunately, beyond simple summaries such as histograms of user ratings, there are few analytic tools that can provide insights into user reviews. In this paper, we propose WisCom, a system that can analyze tens of millions user ratings and comments in mobile app markets at three different levels of detail. Our system is able to (a) discover inconsistencies in reviews; (b) identify reasons why users like or dislike a given app, and provide an interactive, zoomable view of how users' reviews evolve over time; and (c) provide valuable insights into the entire app market, identifying users' major concerns and preferences of different types of apps. Results using our techniques are reported on a 32GB dataset consisting of over 13 million user reviews of 171,493 Android apps in the Google Play Store. We discuss how the techniques presented herein can be deployed to help a mobile app market operator such as Google as well as individual app developers and end-users.

Categories and Subject Descriptors: H.2.8 Database applications: Data mining

Keywords: mobile app market; user rating and comments; text mining; sentiment analysis; topic model.

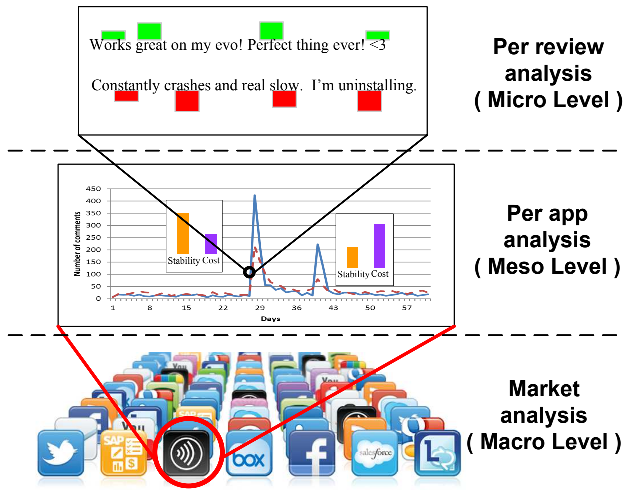
Works great on my evo! Perfect thing ever! <3
Constantly crashes and real slow. I'm uninstalling.
Per review analysis (Micro Level)
Number of comments
Stability Cost
Stability Cost
Days
Per app analysis (Meso Level)
Market analysis (Macro Level)

Figure 1: Three-level analysis in WisCom. Per review analysis (Micro level, Section 4): identifying sentiments and their strength in each review; Per app analysis (Meso level, Section 5): uncovering main causes of user complaints and their evolution over time for each app; Market analysis (Macro level, Section 6): discovering global trends in the whole marketplace.

1. INTRODUCTION

The proliferation of smartphones is driving the rapid growth of mobile app stores. As of this writing, Google Play Store, the official and the largest Android app repository, offers over 700,000 mobile apps [1] mostly developed by third-party companies, organizations and individual developers. User reviews on mobile app stores differ from other online stores in two significant aspects: (a) these reviews are generally shorter in length since a large portion of them are submitted from mobile devices on which typing is not so easy; (b) individual app may have multiple releases, therefore reviews are often specific to a particular version and vary over time. Like other mobile app markets, Google Play displays histograms of ratings and lists text comments by users, in addition to the app's descriptions as submitted by its developers. We introduce techniques for summarizing and mining these reviews and discuss how these techniques can benefit different parties: (a) End-users can use these summaries to choose the apps with the best user experience, without having to read every comment; (b) App developers can use these summaries to understand why end-users love or hate their apps, as well as competing apps, so that they can improve their quality; and (c) Market operators such as Google Play can use our techniques to automatically spot problematic apps to ensure safe and quality content and steer the market towards greater prosperity. (d) All these and other interested parties can benefit from the analysis of market segments and trends.

While one could manually analyze these aspects, it is extremely tedious due to the sheer quantity of ratings and comments. There are few fast and reliable tools for users or developers to quickly grasp the main ideas and primary concerns from a large number of reviews. For example, the popular game Plants vs. Zombies has a rating of 3.6 out of 5, but short of reading several hundred comments, it's difficult to gauge what the major complaints people have about this app. Even worse this rating is averaged over multiple releases, which is not very useful in informing users who usually only care about the latest release.

Towards this end, we propose WisCom, a multi-level system that analyzes user reviews at various granularities. More specifically, our system provides analysis on single review (micro level), reviews of each app (meso level), and all apps in the market (macro level), as shown in Figure 1.

In the micro level analysis, we target individual text comments and perform word-level analysis to understand the impact of each word on users' actual sentiments. This analysis helps us identify the vocabulary users used to praise or criticize apps. By applying a regularized regression model, we are also able to predict the rating score based on the text comment users posted. One interesting application of micro analysis is that it helps us detect ratings that do not match the actual text of the comments. We found this type of inconsistency in roughly 0.9% of the user reviews we collected. These inconsistencies may be caused by careless mistakes or indicative of intentionally misleading reviews (e.g. reviews entered by competitors, or possibly by the developers themselves to boost their app rating).

In meso level analysis, we aggregate comments of individual apps and use text analysis tools (such as Latent Dirichlet Allocation [6]) to further study why users dislike these apps[1]. By performing topic analysis on different time segments, we are able to provide a dynamic view of how users' opinions evolve over time, therefore to discover event-driven trends, and life spans of different versions.

We further extend our analysis to the scope of whole marketplace in the macro level analysis. We are aiming at understanding the general user preferences and concerns over different types of apps and providing guidelines to developers or even market operators. For example, our findings suggest that for paid application apps, users are more concerned about its cost, whereas for paid games, users' concerns lie in other factors such as stability and attractiveness. Macro level analysis can potentially be used in market analysis to unveil underlying patterns, and answer various questions relevant to different stakeholders, ranging from "What are the most important qualities users care about in mobile games?" to "Which types of apps users are more willing to spend money on?".

We believe our work offers important insights that benefit end-users, developers and potentially the entire mobile app ecosystem. More specifically, we provide techniques and tools that allow people to easily absorb information contained in large set of text reviews and numerical ratings by offering multiple forms of summarization. Our contribution can be summarized as follows:

- Our proposed WisCom can automatically summarize and make sense of user reviews at micro, meso, macro levels. WisCom is able to
  - detect inconsistent comments/ratings;
  - identity root causes of users' negative reviews;
  - track the evolving pattern of users reviews;
  - discover market trends
- We collect and study a 32.4 GB dataset that consists of more than 13 million user reviews of 171 thousand Android apps in the Google Play Store.

[1]The same technique can also be used to analyze why people like an app, but in this paper we focus on negative reviews since those can be directly used to improve app quality.

2. RELATED WORK

Thus far, there has been little work in mining app store data. Most of the past work here has focused on the apps rather than their user reviews, though. For example, Frank et al. crawled a corpus of 188,389 Android apps from several Android app stores including the official Google Play Store [9]. Their objective was to uncover the patterns in the Android permission requests by applying boolean matrix factorization rather than analyzing the user reviews.

Topic models [6, 5, 17, 18] have been widely used to find meaningful topics (i.e. clusters of words) from text or image corpus. Hong and Davison performed an empirical study of Twitter messages [12]. Since the messages are often short on Twitter, they proposed to train a topic model on aggregated messages to achieve better performance. In our work, the user reviews are also short and standard topic models do not apply well on single review, therefore we also concatenate the user reviews. Blei and Lafferty [5] proposed the dynamic topic models in which they used state space models on the natural parameters of multinomial distributions that represent the topics. In our system, we adopt a different approach to first identify peaks in number of comment streams and then to analyze the topics, which we call "root causes".

There are several pieces of past work analyzing reviews of other kinds of marketplace, such as markets of tangible commodity goods and movies [15, 7, 19, 4, 8, 11]. Hu and Liu [13] provided a feature-based summary of a large number of customer reviews of products and extracted opinion sentences to perform sentiment analysis. Archak et al. [4] used techniques that decompose the reviews into segments that evaluate the individual characteristics of a product such as quality, price and etc. They found that customers place different weight on each individual product features for different products. In our work, we found similar pattern that users have different concerns over different types of mobile apps. There were successes in applying word-level regression to movie review to predict a movie's opening weekend's revenue [15], and to food menus to correlate dishes' prices [7]. Their techniques rely on textual features, and cannot be directly applied to mobile app stores since they often have different review styles. For example, the reviews on movie and commodity websites tend to be longer, while those of mobile apps are often short (average 71 characters per comment).

Much work has focused on detecting spam reviews [14, 16, 20, 22, 21]. This line of work focuses on detecting and removing fraudulent reviews to provide a fairer marketplace. In our work, inconsistent review detection can also help identify fraudulent reviews (e.g. reviews posted by competitors using Sybil attacks) though our major objective is to remove nonsensical comments, and to improve the performance of root causes discovery.

Table 1: Description of app's meta-data and user review

Attribute	Description
	Meta-data
App Name	The name of the app
Category	30 app categories defined in Google Play, e.g. Racing, Business, Tools etc.
Price	Cost of the app in US Dollars
Content rating	Suitable audience for the content
Downloads	Number of Downloads, e.g. 1-5, 5-10, ..., 100,000,000-500,000,000
Avg rating	Average rating received by this app
Rating dist	Number of 1-star, 2-star, ... , 5-star ratings
Review count	Number of reviews submitted[4]
	User review
App Name	Name of the reviewed app
Timestamp	Unix timestamp of the creation time
Rating	The rating score given to this app on a 1-5 scale
Comment	The comment text entered by the reviewer

3. OVERVIEW OF THE MOBILE APP STORE DATA

In this section, we describe how we obtained our dataset, how the attributes are structured and the basic descriptive statistics of the dataset.

3.1 Data Collection

We collected meta-information and user reviews of 171,493 Android apps[2] from Google Play in November 2012. Each Android app in Google Play has its own description page. However, there is no index of all of the publicly available apps. To build our dataset, we ran a Breadth-First-Search starting from Google Play's home page, and crawled all of the web pages containing app description information. Once we got a description page, we parsed the HTML page to extract the app's metadata, including its name, category, number of downloads[3], average user rating score, rating distribution, price, and content rating. All the attributes and possible values of app's meta data are summarized in Table 1.

Next, for each app we crawled all the user reviews through an open-source Google Play API [3]. We crawled a total of 13,286,706 user reviews. Each user review consists of a timestamp showing when the review was created, a user rating, and the user's comment in the text form, all of which are also summarized in Table 1. Due to Google's privacy protection measures, we were not able to get the unique identifier of each user who posted comments. Accordingly, the techniques described in this paper do not require information about the identity of each reviewers. Together with the apps' metadata, it takes up approximately 32.4 GB of storage when organized in a MySQL database.

[2]Google defines two major categories for the programs in their market, "game" and "application". Throughout this paper, we use "app" to refer to the programs that users can download to their smartphones, and "application" to refer to the category of apps that are not games.

[3]Google does not provide the absolute number of downloads. Instead, it discretizes this number into several ranges.

[4]For apps having more than 6000 reviews, we only crawled its most recent 6000 reviews.

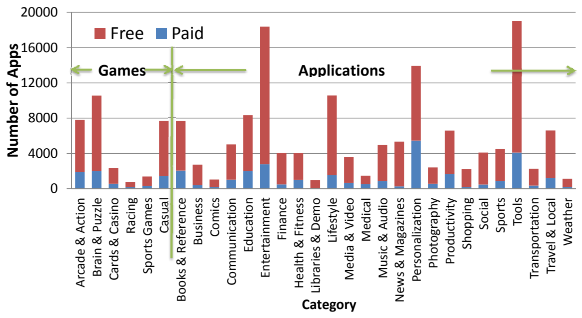

Figure 2: Statistics of free and paid apps in each category.

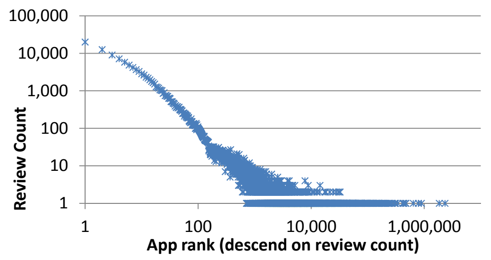

Figure 3: The distribution of comments. The number of comments follows heavy tailed distribution.

3.2 Descriptive Statistics

All the apps we collected belong to one of 30 pre-defined Google Play categories. Of the 171,493 apps we gathered, 136,086 (or 79%) were free. The number of free and paid apps in each category is shown in Figure 2. The percentage of each category is similar to the stats reported in AppBrain [2] back in November 2011. Entertainment, Tools, and Personalization are the top 3 most popular categories. Personalization is also the category with the highest percentage of paid applications. On average, each app in our dataset has 101.90 user reviews (standard deviation=460.25, median=8). The number of reviews roughly follows a heavy tailed distribution, as shown in Figure 3. Google Play's rating system is on a 1-to-5 scale, where 5 stars mean most satisfactory. The breakdown of ratings is shown in Figure 4, where over 54% of the ratings are 5 stars. The average rating over all apps is 3.90 with standard deviation of 1.48. We use these ratings as the labels of each text comment. More specifically, we treated the 3 stars rating as the threshold to distinguish whether users liked or disliked an app, which yields approximately 2.7M negative reviews and 9.5M positive reviews.

4. MICRO ANALYSIS: DETECTING INCONSISTENT REVIEWS

As the first step of our analysis, we analyze individual comments. We want to quantitatively determine if users are praising an app or complaining about it. We built a regression model on the vocabulary users used in their reviews to

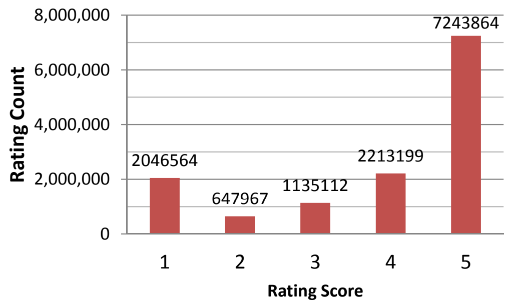

Figure 4: Breakdown of ratings associated with the user reviews. The average rating is 3.90, the median is 5 and the standard deviation is 1.48. Note the U-shape and skewness in the histogram.

conduct comment-level sentiment analysis. For each word, the model gives a numeric weight that measures its average influence on a rating. One important application of this regression model is to identify inconsistencies between text and rating, i.e., Text-Rating-Inconsistency (TRI). We speculate that these comments can be attributed to a combination of careless reviewers and attempts to manipulate ratings. Our analysis offers a powerful tool to detect TRI, and provides more accurate ratings for marketplace and users. As a by-product of this first analysis, we also identified words indicative of negative sentiments. In the next section we will use these words to help differentiate between different categories of user complaints.

4.1 Data Processing

Instead of directly dumping data into a regression model, some data pre-processing is necessary. We sampled one million user comments in our dataset (8% of the total comments), and applied the following steps to clean up review text and decompose user comments into words:

- Remove all the HTML tags;
- Filter out non-English reviews by removing comments that contain more than 5 non-ASCII characters;
- Split the comment strings to words using space and the following delimiters: `. , : ( ) / [ ] ! * ; " ' +`
- Convert all the letters to lower case;
- Remove uncommon words, i.e. words appearing less than 10 times in the entire sample.

After the above pre-processing steps, we harvested 13,674,405 words from 988,960 comments. The vocabulary consists of 19,387 distinct words. This is formalized into a $m \times n$ matrix X, where m is the number of comments, and n is the size of vocabulary, i.e. $m = 988,960, n = 19,387$. X_{ij} indicates the frequency of the j-th word in the i-th comment. We use a vector Y with m elements to represent the ratings. Y_i is the rating given to the i-th comment. The matrix X is very sparse: only 0.07% of the elements are non-zero.

4.2 Regression Model

We applied a linear regression model to model the relationship between review text and rating. Specifically, we trained a linear model with a set of parameters $W = w_0, w_1, ..., w_n$, that minimize the quadratic loss of data:

$$L(W) = \Sigma_{i=1}^{m}(Y_i - (w_0 + \Sigma_{j=1}^{n} X_{ij} w_j))^2 + \gamma P(W)$$

where the regularization term $P(W)$ is defined as:

$$P(W) = \Sigma_{i=1}^{n}((1-\alpha)w_i^2 + \alpha|w_i|)$$

$P(W)$ contains two terms. The first term is the normal Tikhonov regularization that reduces over-fitting. The other is l_1 norm, which produces a large proportion of zero weights. This property is consistent with our case, since most words in the vocabulary should have no or little effect towards rating. As a secondary benefit, l_1 norm controls the size of model to accelerate training and testing, while also serving as a feature-selection job.

To implement this model, we utilized the `glmnet` package in R [10]. It uses cyclical gradient descent algorithm to quickly solve the optimization problem[5]. We ran 10-fold cross validation to find the optimal γ that determines the weights of the regularization terms. The best γ was used to train a regression model on the full dataset. We did not optimize on α, and set it to 1 in cross validation, and 0.2 in the final model.

4.3 Experiments

In this subsection we analyze the results obtained from the linear regression model. We first checked the words that receive the largest positive weights and largest negative weights. Most of them are typos, less-used slangs, or words in other languages with very strong feelings. For example, "parfait", "perfetto", "pooooor", and "suks".

In Table 2(a), we list the words with the largest positive weights (minimum of 1000 appearances). In Table 2(b), we list the words with the largest negative weights (minimum of 1000 appearances). The results are reasonable, but most of these words only express strong emotion without providing specific reasons. What we really want, though, is to identify the weights of words with more informative meanings (like "boring" in Table 2(b)), that can help explain why users give high/low ratings.

Table 2: Words with the outstanding positive weights (a) and negative ones (b).

(a)

Word	Weight	Freq
awsome	0.67	4893
excellent	0.67	31971
awesome	0.63	63257
fault	0.61	1027
sweet	0.60	3572
superb	0.58	3694
brilliant	0.58	6384
yay	0.57	1134
greatest	0.56	1148
amazing	0.56	18753

(b)

Word	Weight	Freq
sucks	-1.24	13178
lame	-1.16	2701
rubbish	-1.12	2127
worthless	-1.03	1628
poor	-1.02	6307
boring	-0.99	3529
useless	-0.98	8075
horrible	-0.96	4428
crap	-0.95	7515
garbage	-0.93	2217

The negative weights of different words, especially words that link to specific features of apps, can be used to imply how different types of defects weigh in users' perceptions. As shown in Table 3, we list the weights of 10 words that convey several common complaints users have. For example, "bloatware" receives the largest negative weight in this table.

[5]The complexity of the algorithm in each iteration is linear in the total number of words in the corpus.

Table 3: Examples of negative words that can imply the problems with an app.

Word	Weight	Freq	Word	Weight	Freq
bloatware	-0.89	1778	slow	-0.44	9939
misleading	-0.84	465	confusing	-0.38	1300
crashes	-0.71	9081	expensive	-0.26	1538
spam	-0.62	1601	permission	-0.18	1409
freezes	-0.54	3960	privacy	-0.10	962

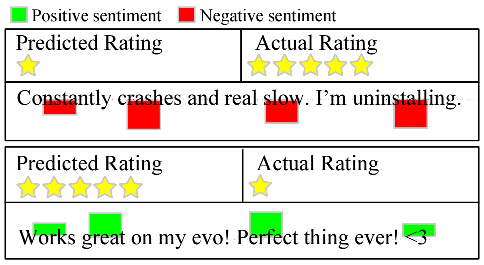

Figure 5: Two examples of TRI identified by WisCom. Note the discrepancy between users' ratings and their comments, while our predicted ratings match the comment text.

On average, each use of this word in a comment reduces the rating by 0.89. Words like "permission" and "privacy", although also negative, tend to be associated with milder sentiment. This may suggest that users are more bothered by memory-hungry apps than by those requesting excessive permissions.

4.4 Discovering Inconsistent Reviews

One application of this model is to detect comments with Text-Rating-Inconsistency. Figure 5 illustrates two examples of TRI. The first comment expresses strong negative sentiment, but is associated with a 5-star rating. The second example provides an opposite case. Our model accurately caught these inconsistencies.

We applied the regression model on a separate test set with 50,000 comments, and checked the differences between the actual ratings of comments and the predictions. Among these testing comments, we identified 0.9% as being inconsistent, namely the distance between actual rating and predicted rating is greater than or equal to 3. Table 4 illustrates more examples of TRI that we found.

Through manual inspection, we were able to determine that the vast majority of comments identified as TRI were indeed inconsistent. Some TRI comments are probably careless mistakes from users, while others may indicative of attempts to manipulate ratings. Either way, our model provides an automatic approach to detect TRI and also gives more accurate ratings. We believe both marketplace and users can benefit from this analysis. For instance, the marketplace could automatically remove these comments and/or exclude them from the computation of average ratings. Alternatively, it could also flag them to alert users about the inconsistencies. Removing these TRI reviews also helps us reduce the noise in the dataset, yielding better performance in later analysis.

5. MESO ANALYSIS: DYNAMIC VIEW OF ROOT CAUSES

In the previous section we presented the per review analysis, which helps us better understand each individual comment and the words in it. However, knowing whether users are praising or criticizing an app is definitely not enough. Instead we would like to be able to automatically inform app market operators, developers and users about the specific problems behind the complaints reported by users in their reviews. We refer to this as "root cause analysis" (of the complaints).

5.1 Topic Analysis

To identify meaningful root causes, we applied the Latent Dirichlet Allocation model [6], one of the most widely used topic model nowadays, to analyze user reviews. More specifically, we discovered topics that correspond to the root causes of people's concerns toward apps. For each app, we analyzed its topic distribution, and found out the strongest complaints users have. Moreover, we combined topic modeling with time-series plots, providing a dynamic view over time. Users can utilize this tool to better understand the outstanding characteristics of an app throughout its life span.

5.1.1 Topic Model Experiments

In addition to the data pre-processing we conducted in Section 4.1, three major steps were added. First, we removed all the inconsistent reviews to filter out noise in the data. Second, to spot popular reasons why users are unsatisfied with certain apps, we only chose *negative* comments, which are associated with 1-star or 2-star ratings. Third, to better focus on the users' negative sentiment, we filtered out words that receive a non-negative weight in the linear regression model mentioned in Section 4.

Compared to other types of documents, most user comments are relatively short. Average length of the comments is 71 characters, and median length is 47 characters. After filtering out the non-negative vocabulary, the comments in the resulting corpus are even shorter. Therefore, we chose to concatenate comments from the same app together as a new document. Furthermore, we filtered out documents that are less than 100 characters. The resulting corpus contains comments from 52,631 apps.

To train an LDA model, we used the Stanford Topic Modeling Toolbox[6]. Table 5 illustrates the result of a 10-topic LDA model. For each topic, the top-10 weighted words in its vocabulary distribution are listed. The topics are sorted by their average proportions across the distribution of all documents. We add a descriptive word to each topic at the top of Table 5 to represent the major concept each topic is talking about. Most topics exhibit clear reasons why users dislike an app. These reasons relate to functional features such as picture and telephony, performance issues such as stability and accuracy, and other important factors such as cost and compatibility. For each complaint category, we give examples of representative apps that suffer most from relevant problems. For example, we found that most complaints users have about the mobile game StarDunk and Blast Mon-

[6]http://nlp.stanford.edu/software/tmt/tmt-0.4/. This toolbox uses variational EM for learning. Its theoretical complexity of each iteration is linear in the total number of words, and in the number of topics [6].

Table 4: TRI analysis on the testing comments. We chose the ones whose actual rating is the most different from the output of our model. Check marks (✓) indicate true positive detections. P is prediction, and A is actual rating.

TRI	P	A	Comment text
✓	1	5	Sucks Dont waste yhor time if yho get it yho will be sorry
✓	1	5	Non stop force closes Its done it 5 times while rating this junk app! Refund please!! I wish I could rate it zero stars and stop dev from spam rating his own app! Worthless!!DO NOT BUY THIS APP THERE ARE WAY BETTER ONES FOR WAY LESS OR FREE!!!
✓	1	5	Terrible Subscriptions , all gone. Bull crap . Fix this sheet
✓	5	1	This is awesome. Love it. Works with droid the best.
✓	1	5	Not working on galaxy nexus prime It was working then suddenly stopped uninstaled it an re install still not working any one know how to get it working
✓	1	5	Dish sucks Quit working and dish sucks not doing anything about it
✓	5	1	awesome awesome
✓	1	5	Crap aap waste Haha didnt recoganise any song crap app waste of time
✓	5	1	Best app I've down loaded for my droid X. Super simple to use with great results.
✓	5	1	Best freaken apps on my phone r this one and the other body parts from the same person. Freaken awesome! I deff feel the burn
✓	5	1	Works great on my evo! Perfect thing ever! <3
✓	1	5	Crap iv transfered pics & they vanished iv lost some precious pics !!!! Rubbish
?	5	1	Dg lala lala lala its elmos world
	1	5	Not sure why this app isn't bloat, but its great and should be on every T-Mobile device...I don't understand y u force the crap apps on us but don't force the good ones on us...

keys are related to its unattractive content. Opera Browser is disliked by users mostly because it occasionally crashes (unstable).

5.2 Dynamic Analysis

An important consideration when looking at app reviews has to do with the successive releases associated with most apps. Different releases may suffer from different problems and elicit different complaints. Instead of attempting to identify problems associated with an app by blindly combining comments collected over its entire life span, we opt to segregate comments by releases. To achieve this, we visualize the life span of an app by plotting time series of reviews from its creation to the time last review was posted. We observed that spikes in reviews are closely correlated with new releases of an app. For example, the biggest spike of app Plants Vs. Zombies (Figure 6) appeared right after this game entered the Google Play Store on Dec. 21st, 2011.

Spikes come primarily in the form of bursts of positive or negative comments. There are also situations where a positive spike shortly follows a negative spike, an indicator of a quick fix to a problem introduced in a new release. In this section, we use the Plants vs. Zombies game as an example to illustrate how time series and root cause analysis help us recreate the history of an app (Figure 6). This game was first released on Google Play on December 21, 2011 (day 1 in the figure). There was a significant burst of negative reviews due to the unstability of the initial release (see Figure 6 (a)). When we applied root cause analysis, we found that a large portion of the reviews were complaining about the stability, as we quote one review on Dec 30 "Fix it! It keeps force closing on stage 1, need an update.... please!!!". Following this initial spike of negative reviews, stability remained the main source of complaints (see Figure 6 (b)) until a follow-on release in May 2012, which fixed the stability problem but resulted in connectivity issues (see Figure 6(c)). The following quote posted on May 30, 2012 illustrates the emergence of this new problem:"Would give 0 stars if I could. Server error. App will not open. 2nd device it will not work on. Want a refund!" Approximately a week after this incident, there was a spike of positive review on the time series plot, containing reviews such as "Finally fixed. Hooray, no more crashing. Thanks, now for zombie killing. >:)". These and other reviews indicate that the connectivity problem had been solved. It also explains the quick drop in negative reviews around June 6th.

This dynamic analysis gives us a historical view of apps, which is extremely useful for users to gain a deeper understanding of apps. In short, our analysis can be used to not just alert app market operators, developers and users about potential problems but to help them identify the nature of these problems.

6. MACRO ANALYSIS: HIGH-LEVEL DISCOVERIES OF MARKET TRENDS

In this section, we extend our analysis to the entire app market, trying to answer two additional questions:

- What are the most critical aspects the app developers should pay attention to when developing different types of apps?
- Whether users have the same expectation and tolerance for applications as for games?

The high-level trends we discover offer great lessons to developers and can be used to improve the market efficiency if used properly.

6.1 Outstanding Complaints in Each Category

We have summarized the top-10 complaints from users' negative reviews identified in the previous section. A follow-up question one may ask is whether users have similar complaints on different apps. Our intuition tells that this may not be the case, since different types of apps utilize different features and serve different purposes. Our data seems to

Table 5: Most frequent words from the top 10 causes found by WisCom–topic model.

cause	*Attractiveness*	*Stability*	*Accuracy*	*Compatibility*	*Connectivity*	*Cost*	*Telephony*	*Picture*	*Media*	*Spam*
Words	boring	closes	find	galaxy	log	free	uninstall	pictures	video	ads
	bad	close	location	battery	error	money	want	picture	sound	notification
	stupid	load	search	support	account	buy	need	pics	watch	spam
	waste	every	info	off	connect	pay	send	camera	videos	bar
	dont	crashes	useless	droid	login	paid	messages	save	songs	notifications
	hard	keeps	data	nexus	connection	refund	delete	wallpaper	audio	adds
	make	won	way	compatible	sign	want	let	see	sounds	annoying
	way	start	list	install	let	back	contacts	photos	hear	many
	graphics	please	sync	samsung	slow	bought	calls	upload	record	pop
	controls	closing	wrong	worked	website	waste	off	pic	anything	push
%	18%	13%	13%	11%	10%	9%	8%	8%	5%	5%
Example app	Stardunk	Opera	Kindle	App 2 SD	Zedge	Sygic	LINE	Pho.to Lab	IMDB	Brightest Flashlight
	Blast Monkeys	Bible	Kobo	Solar Charger	Dropbox	Cut the Rope	WhatsApp	Retro	Tuner	Shoot the Apple

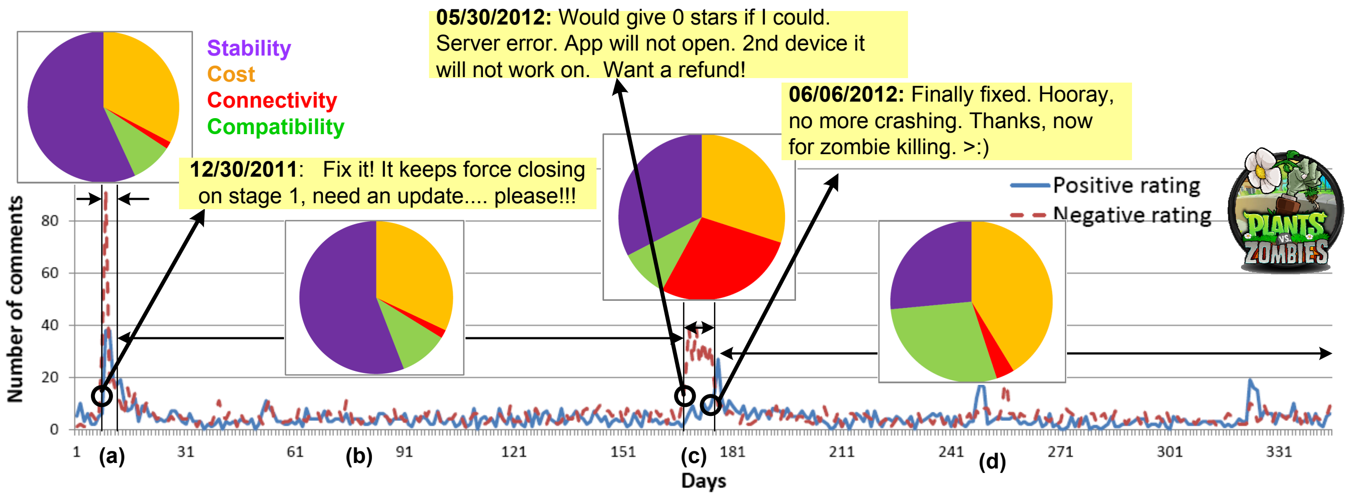

Figure 6: Per app analysis in WisCom. We use time series to visualize the life story of Plants vs. Zombies, and WisCom performs topic analysis for different segments of the time series.

confirm this point. For each app, we determine its most common complaints, and aggregate them on categories. Here we list the top-3 complaints in each category as shown in Table 6 with the numbers indicating the proportion of apps involved with each complaint. For example, 60% of the Arcade & Action games are criticized most as unattractive, 18% of them suffer most from stability problems, and 11% disliked mainly because of their costs.

Surprisingly, for all seven categories of games, the top-3 complaints revolve around the same issues: content attractiveness, stability, and cost. The attractiveness takes a significant weight among these three aspects, which suggests that content of a game is a key success ingredient. On the other hand, users complained about different things for different categories of applications. For example, complaints on accuracy stands out in Book & Reference, Lifestyle, Productivity, Transportation, Travel, and Weather categories, whereas in Business, Finance, Social, and Sports categories the most common complaint has to do with connectivity. The source of complaints for different categories of apps is an indication of which factors seem to matter most in different app categories. Therefore, developers should take notice of these aspects to make their products more appealing.

6.2 User Reception of Games and Applications

In this section we provide in-depth analysis on two major categories of apps, games versus applications. We show that (a) Applications usually receive more unified complaints, where the dissatisfaction of a game can be attributed from multiple reasons; and (b) Users are more tolerant to the cost of mobile games than applications.

We visualize the distribution of complaints on ternary plots (Figure 7). In this figure, we focus on three common areas of complaints: unstable, unattractive and costly. We only consider the apps whose complaints related to these three reasons take up at least 50% of all complaints they received. Among them, We display the 100 most reviewed free applications and free games on the left, and the 100 most reviewed paid applications and paid games on the right. When we look at the two ternary plots separately, we observe that among all the free apps on the left, applications exhibit more polar complaints. A significant portion of these applications

Table 6: Top-3 complained aspects of each app category.

	Category	1st Complaints	2nd Complaints	3rd Complaints
Game	Arcade & Action	Attractiveness (60%)	Stability (18%)	Cost (11%)
	Brain & Puzzle	Attractiveness (49%)	Stability (18%)	Cost (8%)
	Cards & Casino	Attractiveness (41%)	Cost (23%)	Stability (19%)
	Racing	Attractiveness (61%)	Stability (14%)	Cost (11%)
	Sports Games	Attractiveness (65%)	Stability (15%)	Cost (10%)
	Casual	Attractiveness (54%)	Stability (17%)	Cost (8%)
Application	Books & Reference	Accuracy (26%)	Phone (13%)	Connection (13%)
	Business	Connection (31%)	Accuracy (22%)	Cost (15%)
	Comics	Attractiveness (29%)	Picture (17%)	Connection (16%)
	Communication	Phone (33%)	Connection (18%)	Compatibility (13%)
	Education	Attractiveness (17%)	Accuracy (13%)	Phone (13%)
	Entertainment	Attractiveness (28%)	Media (16%)	Stability (11%)
	Finance	Connection (43%)	Accuracy (25%)	Cost (9%)
	Health & Fitness	Accuracy (38%)	Stability (11%)	Attractiveness (10%)
	Libraries & Demo	Attractiveness (21%)	Compatibility (19%)	Phone (15%)
	Lifestyle	Accuracy (26%)	Stability (12%)	Connection (12%)
	Media & Video	Media (28%)	Picture (15%)	Stability (12%)
	Medical	Accuracy (30%)	Cost (19%)	Connection (12%)
	Music & Audio	Media (32%)	Stability (17%)	Attractiveness (11%)
	News & Magazines	Connection (40%)	Stability (19%)	Accuracy (12%)
	Personalization	Picture (29%)	Compatibility (19%)	Spam (13%)
	Photography	Picture (61%)	Stability (10%)	Cost (6%)
	Productivity	Accuracy (31%)	Compatibility (16%)	Connection (15%)
	Shopping	Accuracy (54%)	Connection (16%)	Stability (12%)
	Social	Connection (34%)	Phone (13%)	Stability (12%)
	Sports	Connection (25%)	Accuracy (21%)	Stability (16%)
	Tools	Compatibility (29%)	Accuracy (16%)	Phone (13%)
	Transportation	Accuracy (52%)	Cost (11%)	Stability (9%)
	Travel & Local	Accuracy (50%)	Connection (11%)	Cost (9%)
	Weather	Accuracy (56%)	Compatibility (11%)	Stability (10%)

scattered close to the corners of the triangle[7]. This implies that many applications only have one complaint stands out, whereas in games, complaints are mixed from all three aspects. Paid apps (Figure 7(b)) show the same characteristic, too.

When we compare the two plots horizontally, we notice that the differences in the distribution between free applications and paid applications is much more dramatic than that of free and paid games. There is a significant portion of paid applications that receive very strong complaints about their prices, but much less paid games do. In other words, users seem to be more tolerant to the costs of mobile games. They are generally more willing to spend money on high quality games than on high quality apps.

Although this analysis is specific to the differences between games and applications, the same method can be extended to other features and other subcategories on the app market.

7. CONCLUSION

In this work, we collected and studied over 13 million user reviews from Google Play. We proposed WisCom, an integrated system to analyze user reviews from three different levels, namely comment-word centric analysis, app centric analysis, and market centric analysis. Our system was able to detect the inconsistencies between user comments and ratings, identify the major reasons why users dislike an app, and learn how users' complaints changed over time. We also extended our analysis to the scope of the entire marketplace, discovering high-level knowledge and global trends in the market. WisCom greatly improves the existing feedback channel in mobile app markets, benefiting end-users, app developers, market operators and other relevant stakeholders in mobile app ecosystem.

In our future work, we will use WisCom to analyze other secondary Android markets as well as other online marketplaces. We will also apply more in-depth analysis to investigate the different review patterns triggered by various market operations or external events and to what extend these patterns can be used in market prediction.

[7]Although these apps are free, we still see users complaining about their cost. This is possibly because of that some applications have two versions, a free version and a paid premium version. Users of the free version sometimes complain about the costs of the premium version.

8. ACKNOWLEDGMENTS

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 and W911NF-09-1-0273, by CyLab at Carnegie Mellon under the grant DAAD19-02-1-0389, by NSF CNS-101276, CNS-0905562, and CNS-1228813, by DARPA FA8650-11-1-7153, by Carnegie Mellon Portugal ICTI, and by Google. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either ex-

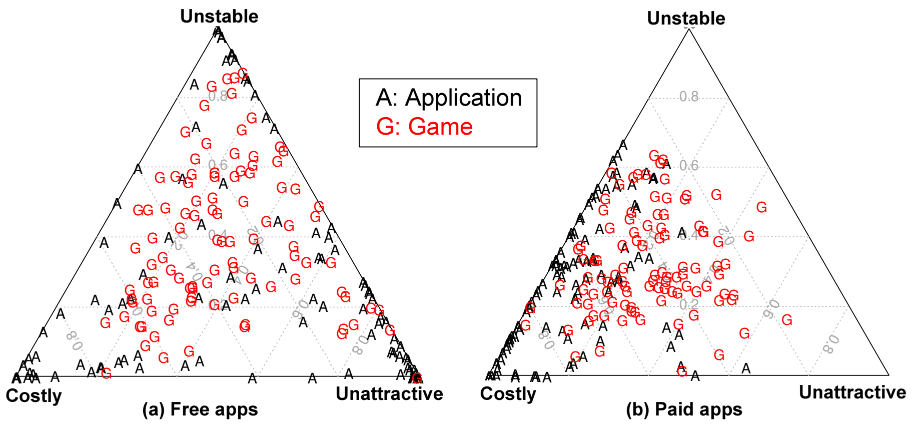

Figure 7: Ternary plots of Application vs. Game on three topics. Each label represents an app. We choose the apps whose majority (>50%) complaints are related to these three topics. The 100 most reviewed apps in each of the cases (free-Application, free-Game, paid-Game, paid-Application) are plotted. Note users complain more about cost for paid applications.

pressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. The authors would like to thank Chong Wang and Miaomiao Wen for their insights in discussions.

9. REFERENCES

[1] Google play has 700,000 apps, tying apple's app store, http://news.yahoo.com/google-play-700-000-apps-tying-apples-app-172110918.html.

[2] Most popular android market categories, http://www.appbrain.com/stats/android-market-app-categories.

[3] An open-source api for the android market, http://code.google.com/p/android-market-api/.

[4] N. Archak, A. Ghose, and P. G. Ipeirotis. Deriving the pricing power of product features by mining consumer reviews. *Management Science*, 57(8):1485–1509, 2011.

[5] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML*, pages 113–120, 2006.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[7] V. Chahuneau, K. Gimpel, B. R. Routledge, L. Scherlis, and N. A. Smith. Word salad: Relating food prices and descriptions. In *EMNLP-CoNLL*, pages 1357–1367, 2012.

[8] X. Ding, B. Liu, and P. S. Yu. A holistic lexicon-based approach to opinion mining. In *WSDM*, pages 231–240, 2008.

[9] M. Frank, B. Dong, A. P. Felt, and D. Song. Mining permission request patterns from android and facebook applications. *ICDM*, 0:870–875, 2012.

[10] J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2 2010.

[11] A. Ghose and P. Ipeirotis. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *TKDE*, 23(10):1498–1512, 2011.

[12] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *SOMA*, pages 80–88, 2010.

[13] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, pages 168–177, 2004.

[14] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, pages 219–230, 2008.

[15] M. Joshi, D. Das, K. Gimpel, and N. A. Smith. Movie reviews and revenues: an experiment in text regression. In *HLT*, pages 293–296, 2010.

[16] F. Li, M. Huang, Y. Yang, and X. Zhu. Learning to identify review spam. In *IJCAI*, pages 2488–2493, 2011.

[17] C. Lin, Y. He, C. Pedrinaci, and J. Domingue. Feature lda: A supervised topic model for automatic detection of web api documentations from the web. In *ISWC*, pages 328–343. 2012.

[18] D. M. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *UAI*, pages 411–418, 2008.

[19] A. Mukherjee and B. Liu. Modeling review comments. In *ACL*, pages 320–329, 2012.

[20] A. Mukherjee, B. Liu, and N. Glance. Spotting fake reviewer groups in consumer reviews. In *WWW*, pages 191–200, 2012.

[21] G. Wang, S. Xie, B. Liu, and P. Yu. Review graph based online store review spammer detection. In *ICDM*, pages 1242 –1247, 2011.

[22] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In *KDD*, pages 823–831, 2012.