Noname manuscript No.
(will be inserted by the editor)

Analyzing and Automatically Labelling The Types of User Issues that are Raised in Mobile App Reviews

Stuart McIlroy, Nasir Ali, Hammad Khalid and Ahmed E. Hassan

the date of receipt and acceptance should be inserted later

Abstract Mobile app reviews by users contain a wealth of information on the issues that users are experiencing. For example, a review might contain a feature request, a bug report, and/or a privacy complaint. Developers, users and app store owners (*e.g.*, Apple, Blackberry, Google, Microsoft) can benefit from a better understanding of these issues – developers can better understand users' concerns, app store owners can spot anomalous apps, and users can compare similar apps to decide which ones to download or purchase.

However, user reviews are not labelled, *i.e.*, we do not know which types of issues are raised in a review. Hence, one must sift through potentially thousands of reviews with slang and abbreviations to understand the various types of issues. Moreover, the unstructured and informal nature of reviews complicates the automated labelling of such reviews.

In this paper, we study the multi-labelled nature of reviews from 20 mobile apps in the Google Play Store and Apple App Store. We find that up to 30% of the reviews raise various types of issues in a single review (*e.g.*, a review might contain a feature request and a bug report). We then propose an approach that can automatically assign multiple labels to reviews based on the raised issues with a precision of 66% and recall of 65%. Finally, we apply our approach to address three analytics proof-of-concept use case scenarios: (i) we compare competing apps to assist developers and users, (ii) we provide an overview of 601,221 reviews from 12,000 apps in the Google Play Store to assist app store owners and developers and (iii) we detect anomalous apps in the Google Play Store to assist app store owners and users.

Keywords Mobile apps, reviews, multi-labelling, anomaly detection

1 Introduction

The Mobile app market continues to grow at a very rapid pace with thousands of developers, thousands of apps, and millions of dollars in revenue [1]. Mobile apps are available through app stores such as the Apple App Store, the Blackberry World Store, the Google

Stuart McIlroy, Nasir Ali, Hammad Khalid and Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
School of Computing, Queen's University, Canada
E-mail: {mcilroy, nasir, hammad, ahmed}@cs.queensu.ca

Table 1: The number of user reviews occurring in a five day window (Sept 4 to Sept 9 2013).

App Name	User Reviews in a 5 day period
AccuWeather	878
The Weather Channel	665
WeatherBug	298

Play Store, Microsoft Phone Apps Store and many more specialized or regional app stores. Such stores provide a convenient and efficient medium for users to download apps and to provide feedback on their user-experience through mobile app user reviews.

Such mobile app user reviews (from this point forward, we refer to the description section of a mobile app user review as simply a user review) contain valuable information *e.g.*, feature requests, functional complaints, and privacy issues. This information is valuable to mainly three stakeholders (from this point forward, we will use the term "stakeholders" to refer to developers, users, and app store owners): (i) developers receive timely feedback about issues related to their apps *e.g.*, bugs, feature requests, and any other issue, (ii) a user can read user reviews to make an informed decision whether or not to download/purchase an app, and (iii) app store owners, *e.g.*, Apple, Blackberry, Google and Microsoft, can analyze user reviews to uncover anomalous apps, *e.g.*, an app with an unexpected number or type of issues relative to other apps.

Due to the large number of user reviews and their free form, it is infeasible for stakeholders to fully benefit from the valuable information in user reviews through manual inspection. As seen in Table 1, there are hundreds of reviews that may occur per day for popular apps. The valuable information in such reviews has led to the emergence of a whole set of companies - mobile app analytics companies. Such companies specialize in providing detailed statistics and comparative analysis of user reviews and app revenues to their clients *i.e.*, app developers [2, 3]. However, much of the provided analytics are not software engineering oriented yet. For example, the occurrence of words in reviews across competing apps are presented, however, the provided analysis would not link such word occurrences to software related concepts (e.g., software quality).

Automated approaches are needed to automatically label reviews based on the types of raised issues *e.g.*, feature requests, functional complaints, and privacy issues. However, users might raise several issues within a particular review. Figure 1 shows an excerpt of a user review where the user raises three issues about an update issue, the response time, and a functional complaint. We cannot label such user reviews with only a single issue. Moreover, such labelling is a difficult task due to the unstructured nature of reviews with many of them containing slang, lacking punctuation, and containing improper grammar.

In this paper, we present an approach that can automatically assign multi-labels (i.e., multi-labelling) to user reviews. The approach helps the various stakeholders gain an overview of the users' feedback that is readily available in the reviews, provides an overview of an app store and allows for the detection of anomalous apps. We perform a large scale empirical study to answer the following three research questions:

RQ1: How many user reviews contain multiple issue types?

Up to 30% of the user reviews raise more than one issue type, i.e., 30% of the data is multi-labelled. We identify 14 types of issues, e.g., feature requests, functional complaints, and privacy issues. We find that the issue types in our manually-labelled data are skewed towards certain issue types and that some issue types are correlated (i.e., they often co-occur often in the same review).

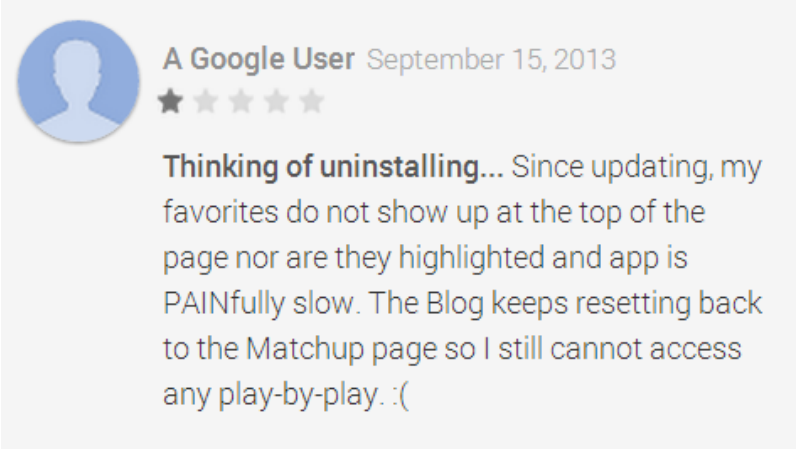

Fig. 1: An example of a Google Play Store one-star user review of 'theScore: Sports & Scores' app. The review contains (raises) multiple issue types

RQ2: How well can we automatically multi-label user reviews?
Our multi-labelling approach can correctly label issue types in user reviews with a precision of 66% and a recall of 65%. We also find that our approach can label some types more accurately (i.e., higher precision and recall) than other types.

RQ3: Are multi-label approaches useful for stakeholders?
We demonstrate several useful applications of our multi-labelling approach for stakeholders using three different scenarios. The first scenario is useful to developers and users when comparing issues across competing apps. The second scenario is useful to app store owners and developers as we analyze the issue distribution of apps in the Google Play Store by app category (e.g., social, finance) and compare the issues from competing app stores for the same app titles. The third scenario is useful to app store owners when detecting anomalous apps based on user reviews and finding apps that violate or disregard the policies and guidelines of an app store (i.e., Google Play Store).

This paper is organized as follows: Section 2 provides the details of our empirical study and presents our approach. Section 3 presents the application of our approach on three analytics use case scenarios. Section 4 discusses the results of applying our approach. Section 5 discusses the threats to the validity of our study. Section 6 presents the related work. Section 7 concludes our work.

2 Empirical Study

In the empirical study section, we present our study design, the results of our preliminary study and our multi-labelled approach.

2.1 Background

In this section we introduce the goals of our study, we provide a brief background about app stores and mobile app analytics. We also motivate our choice to focus our study on negative reviews.

2.1.1 Study Design

The *goals* of our empirical study are to analyze the extent of multi-labelled user reviews, evaluate the effectiveness of automatically labelling multi-labelled user reviews, and present several applications of automated labelling of user reviews for various stakeholders.

The *quality focus* of our empirical study are the evaluation measures that we use to evaluate the performance of our multi-labelling approach. Our *perspective* is that of practitioners (*i.e.*, stakeholders) and researchers interested in user reviews for mobile apps. The *object* of our case study is user reviews from the Google Play Store and the Apple App Store.

2.1.2 App Stores

We choose two of the most popular app stores, the Google Play Store and the Apple App Store. Our criteria for selection is based on popularity (the Google Play Store and the Apple App Store are the top two most popular app stores) and the availability of tools to automatically collect reviews from both app stores (the tool that we used for the Apple App Store has since been made incompatible with an updated version of the store).

The *Google Play Store* is a digital distribution outlet run by Google. Apart from apps, the Play Store sells other digital media, *e.g.*, e-books, movies and music. The Google Play Store has over 1,000,000 apps available as of July 2013 [4].

The *Apple App Store* is the digital distribution outlet for Apple where users can download third-party apps. The apps range from games, productivity apps, social networking and business apps. There are approximately 775,000 apps in the App Store as of July 2013 [4].

There are paid and free apps available in both stores. Apps can be downloaded and updated from the stores. Once downloaded a user can review the app. The number of reviews associated to an app varies depending on the popularity of the app. Some popular apps have over a million user reviews. In our dataset, the top 65 apps with over 100 million downloads have an average of 353,753 user reviews with a maximum of 2,751,345 user reviews for the Facebook app on the Google Play Store.

Reviews in both stores contain a title, a date, a numerical rating between 1 and 5 (where 1 represents a poor app) and a comment section where the user is free to write whatever they wish.

2.1.3 Mobile App Analytics

There are many app analytics companies that specialize in giving developers tools to understand how users interact with developers' apps, how developers generate revenue (in-app purchases, e-commerce, direct buy) and the demographics of app users [2, 3, 5, 6]. These app analytics companies also provide developers with overviews of user feedback and logging crash reports. Google has their own extensive analytics tools for Android developers such as measuring how users are using their app, locating where the users originate from and how they reached the app, tracking how the developer makes money through in-app purchases

and calculating the impact of promotions on the sales of the app [7]. Vision Mobile performed a study based on a survey of 7,000 developers and found that 40% of developers make use of user analytics tools [8] and 18% use crashing reporting and bug tracking. Previous studies also highlight that developers need app analytics tools. For example, Pagano & Bruegge conducted a study on how feedback occurs after the initial release of a product [9]. The authors concluded that there is a need to structure and analyze feedback, particularly when it occurs in large quantity.

2.1.4 Motivation to study one-star & two-star reviews

We select only one-star and two-star user reviews for analysis as we and previous literature make the assumption that one-star and two-star reviews are indicative of negative issues [10] [11]. We decided to focus on negative issues since users and developers are most interested in addressing such issue types. We confirm our groupings of bad and good reviews by running a sentiment analysis tool [12] over all of the bad (one-star and two-star), neutral (three-star) and good (four-star and five-star) reviews of the studied apps. This tool estimates the positive and negative sentiment expressed in the text. As expected, one-star and two-star reviews were given a negative score while four-star and five-star reviews were given a positive score.

We present the results of our empirical study in the following sub-sections. We first present a preliminary study to motivate our work then we present our approach to automatically multi-label user reviews. We follow the results of our empirical study with a discussion of three proof-of-concept scenarios that demonstrate the application of our approach.

2.2 Preliminary Study

We first explore how often do multiple issues occur in user reviews and if so, how many issue types co-occur within a user review. The amount of multi-labelled data informs our decision to use single-labelled or multi-labelled approaches. Single-labelled approaches map user reviews to one label whereas multi-labelled approaches map user reviews to more than one label. One can use multi-labelled approaches on single labelled data, however; multi-label approaches are more complex as the hypothesis space is larger and so multi-labelled approaches would decrease prediction performance unnecessarily. For example, determining whether a user review contains a functional complaint issue or not is an inherently simpler task than determining whether a user review contains a functional complaint, and/or a network problem, and/or a crashing issue. If we use single-labelled approaches on multi-labelled data we will be guaranteed to mis-label the multi-labelled user reviews. Therefore, we carefully consider our choice of approach based on the amount of multi-labelled data that we observe in our preliminary study.

RQ1: How many user reviews contain multiple issue types?

Our process to answer RQ1 is divided into several sections: data collection, our approach to analyze the data, and our results.

2.2.1 Data Collection

We analyze the user reviews of 20 Apple App Store apps and 4 Google Play Store apps. The list of apps can be found in Table 2. We select apps which cover a broad range of

Table 2: User reviews were collected from these 20 apps on the Apple App Store and user reviews from 4 of these apps were collected from the Google Play Store

Name of Apps
Weight Watchers Mobile, Evernote, Hulu Plus, Yelp, Netflix, CNN App for iphone, Farmville by Zynga, Find my iphone, Word Lens, Foursquare, Facebook, Wikipedia Mobile, Adobe Photoshop Express, Last fm, Kindle, Metalstorm Wingman, ESPN Scorecenter, Mint, Epicurious Recipes shopping list, Gmail

Table 3: Data statistics for data collected from the Apple App Store and the Google Play Store

Collected		**Manually Labelled**		**Auto-Labelled**
Apple App Store	Google Play Store	Apple App Store	Google Play Store	Google Play Store
226,797	3,480	6,390	1,066	601,221

categories and have a significant number of user reviews. Half of the apps that we choose have an above average rating, while the other half have a below average rating. We select 4 matching Google Play Store apps from the Apple App Store to examine how issue types from Google Play Store apps differ from those of Apple App Store apps given the exact same apps. After identifying the apps, we built a simple web crawler to automatically collect the user reviews of these apps from the website of each app store.

In total, we downloaded 226,797 one-star and two-star user reviews for the Apple App Store and 3,480 one-star and two-star user reviews for the Google Play store (see Table 3). The discrepancy between the number of user reviews across the app stores is due to the different APIs that Apple and Google exposed for data collection at the time of our study. Apple App Store allows us a user to subscribe to an RSS feed for user reviews, however the Google Play Store only lists the top 500 reviews for each star rating. Hence, the collection of user reviews of the Google Play Store was restricted as no tool existed at the time, to collect the reviews other than crawling the Google Play Store Website.

2.2.2 Approach

In our previous study [13], we manually labelled a statistically representative sample of 6,390 and 1,066 user reviews for the Apple App Store and the Google Play store respectively. Both amounts are above the number required for a statistical sample with a confidence level of 95% and confidence interval of 5%. We labelled only user reviews with a one-star or two-star rating. In total, a graduate student spent approximately 125 hours to manually analyze and label each user review. A faculty member and a post-doctoral fellow, who are not co-authors of this paper, reviewed the labels for consistency. If they disagreed on a label, they took the majority opinion, *i.e.*, in total there were three votes, one from the individual who labelled the data and two from the individuals who verified the data. If the vote was a three-way tie they came to a consensus which occurred for very few reviews.

Our manual labelling process identified issue types from app reviews via manual inspection. Instead of beginning with a set of issues, we analyzed the data to identify common concepts which can be grouped together. We refer to these concepts as issue types. Our selection of issues was motivated by what we believed would be useful for developers and independent of particular apps. There is no doubt that there exist multiple different issues that may or may not be relevant for developers that we do not consider. However, we believe

Table 4: Descriptions of each issue type.

Issue Type	Description
Additional Cost	Complain about the hidden costs to enjoy the full experience of the app
Functional Complaint	Unexpected behavior or failure
Compatibility Issue	App has problems on a specific device or an OS version
Crashing	The app is often crashing
Feature Removal	One or more specific feature is ruining the app
Feature Request	App needs additional feature(s) to get a better rating
Network Problem	The app has trouble with the network *e.g.*, network lag
Other	A review-comment that is not useful or doesn't point out the problem
Privacy and Ethical Issue	The app invades privacy or is unethical
Resource Heavy	The app consumes too much battery or memory
Response Time	The app is slow to respond to input, or is laggy overall
Uninteresting Content	The specific content is unappealing
Update Issue	The user blames an update for introducing new problems
User Interface	Complain about the design, controls or visuals

that the selected issues is a baseline from which future work can be done to explore the remaining issues [13]. Stakeholders are free to select a different set of issues that they feel would be useful to them. Our approach is independent of the selected issues. In total, we identify 13 issue types from the user reviews that we randomly sampled as shown in Table 4. We include a 14th issue type ("other") that covers any user review that does not conform to the 13 issue types.

In our current study, to observe the relationship between issue types, we calculate the percentage of multi-labelled data per issue type *e.g.*, 39% of functional complaints are multi-labelled and the amount of co-occurrence between the various issue types. For example, a functional complaint occurred in 44% of the user reviews that contained a network problem issue. We calculate the word length of the single versus the multi-labelled data to see how they differ. Finally, to study the type of multi-labelled data, we calculate the label cardinality and label density of each review. Label cardinality and label density are commonly used statistics to describe multi-labelled data [14]. These measures are used to quantify the amount of multi-labelling in the studied data set. Let Y be the number of relevant labels, L be the number of all the different label types and N be the size of the dataset.

$$label\ cardinality = \frac{1}{N}\sum_{j=1}^{N} Y_j \tag{1}$$

Label density is the average number of labels in the example divided by the total number of different label types L.

$$label\ density = \frac{1}{N}\sum_{j=1}^{N} \frac{Y_j}{L} \tag{2}$$

2.2.3 Results

We find that the data set is skewed towards certain issue types as seen in Table 5. Functional complaint, feature request and crashing dominate over the other issue types, while issue types such as uninteresting content and resource heavy occur at a much lower frequency.

Table 5: The percentage of issue types in our labelled dataset as well as the probability that a specific issue type co-occurs with other issues *i.e.*,a multi-labelled user review

Issue Type	**Apple App Store**		**Google Play Store**	
	Freq. %	**Multi-Labelled %**	**Freq. %**	**Multi-Labelled %**
Additional Cost	10.8	27.8	1.7	66.7
Functional Complaint	26.7	39	34.1	50.3
Compatibility Issue	2.4	27.7	14.3	11.2
Crashing	16.2	40.3	11.3	56.7
Feature Removal	6	33.4	3	62.5
Feature Request	19.7	28.8	23.1	43.1
Network Problem	10.3	58.1	19.3	65
Other	11.7	0	6.9	0
Privacy and Ethical Issue	2.4	17	0.7	28.6
Resource Heavy	0.4	41.7	6	56.3
Response Time	1.6	64.1	2.3	80.8
Uninteresting Content	0.7	32.6	0.2	50
Update Issue	11.3	74.7	9	78.1
User Interface	4.6	57.4	3.7	56.4

For the Google Play Store user reviews, 30% (317) of the 1,066 user reviews are multi-labelled. The maximum number of labels for a user review is 4; the average number of labels per review is 1.4.

In the Apple App Store user reviews, 22% (1431) of the 6,390 user reviews are multi-labelled. The maximum number of labels for a user review is 5; the average number of labels per review is 1.2.

We observe that the reviews associated with certain issue types are more multi-labelled than other reviews. The multi-labelled column in Table 5 shows how many times an issue type co-occurred with others. As Figure 5 demonstrates 'functional complaint' occurs with many other issue types. 'Update issue' is in multi-labelled data 75% of the time. On the other side of the spectrum, certain issue types co-occur less frequently with others, such as 'privacy and ethical issue' with only 17% co-occurrences. The issue types with the most co-occurring labels are crashing, feature request, functional complaint, network problem, and update issue.

We find that certain issue types co-occur together with increased frequency than others as Figure 2 shows. For example, we observe that the issue type 'crashing' and 'functional complaint' occur frequently with the 'update issue'. Such co-occurrences of issue types demonstrate that issue types are not independent.

As Figure 3 shows, multi-labelled user reviews are longer than single-labelled user reviews. The intuitive reason is that the reviewer has more to say when raising several types of issues in the same review. On average, a user review contains 32 and 33 words for Google Play Store and Apple App Store user reviews respectively. Multi-labelled user reviews are on average 41 words for both Google Play and Apple App Stores.

Both stores show similar patterns. The user reviews of both stores have low label cardinality and label density. The observed values for label cardinality and density in our dataset are similar to prior research on multi-label text datasets [15]. The label cardinality is 1.36, 1.25 and 1.26 for the Google Play Store, Apple App Store and combined respectively. The label density is 0.10, 0.09 and 0.09 for the Google Play Store, Apple App Store and com-

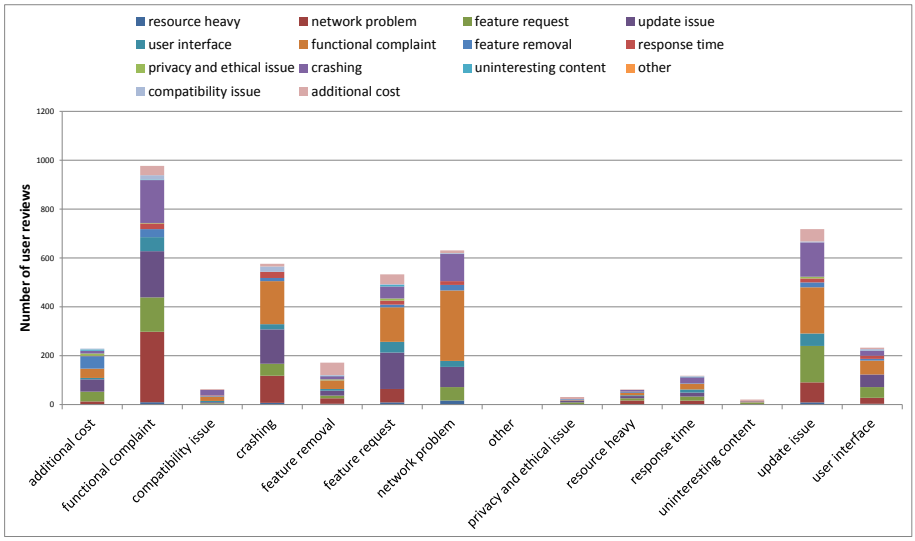

Fig. 2: Amount of co-occurrence per issue types.

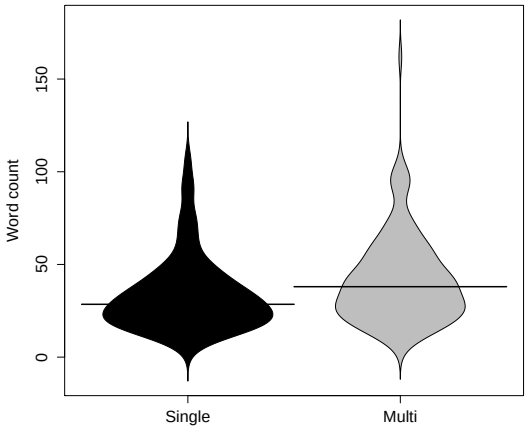

Fig. 3: A bean plot of the length of user reviews in both single and multi-labelled data for both stores

bined respectively. In the following section we make use of prior research techniques for automated labelling of multi-labelled datasets.

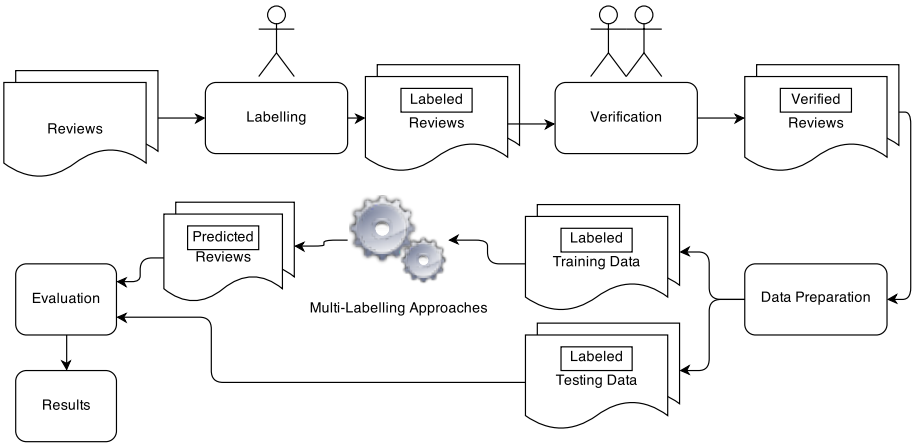

Fig. 4: Process to label multi-labelled user reviews.

30% of Apple App Store and 22% of Google Play Store user reviews are multi-labelled. Users raise multiple issue types in a single review and the distribution of issue types is skewed with some issue types occurring more frequently than others. Moreover, some issue types tend to co-occur at a higher frequency than other issue types.

2.3 Automated Labelling of User Reviews

From the results of RQ1, we conclude that the user reviews of the Google Play Store and Apple App Store contain a substantial amount of multi-labelled data. Hence, we wish to explore how well we can automatically label user reviews using machine learning models that are tailored specifically to multi-labelled data.

RQ2: How well can we automatically multi-label user reviews?

Our process is divided into several steps: data preparation, multi-label approaches and evaluation and results (as shown in Figure 4). In the following subsections, we explain each step in detail.

2.3.1 Data Preparation

The input to the data preparation step is the manually-labelled data from RQ1. The output is a list of processed user reviews.

Preprocessing: For all user reviews, we merge the title and comment together. We choose not to remove stop words because certain issue types benefit from words such as 'should' and 'could' as shown by Iacob & Harrison [16]. We remove all numbers and special characters except hyphens and apostrophes. We filter words that occur less than three times in our data set. Such filtering removes rare instances of misspellings and long sequences of meaningless characters. Filtering words that occur less than three times reduces the complexity of

assigning labels to user reviews and the approaches are less likely to overfit to the data. We stem the words using the snowball stemmer [17] because it implements the popular Porter stemming algorithm [18]. We expand abbreviated words such as 'couldn't' to 'could not'. We ensure that user reviews are meaningful with at least a one-word title and a three-word comment. Hence, we remove any user review that is three words or less *e.g.*, *review: bad!, Not working*. At the end of the preprocessing step, we are left with 7,290 reviews out of the initial set of 7,458 manually labelled user reviews.

Frequency Building: We transform the text of the user reviews into an array of word frequencies. The word frequency format is readable by machine learning algorithms. We use the `String To Word Vector` filter, available in MEKA (a multi-label classification tool [19] which is an extension to WEKA [20]), in order to build a dictionary of all words left after preprocessing. There are 6,525 unique words before preprocessing and 2,771 unique words after preprocessing. The attributes of each user review are the words. The MEKA filter forms a $M \times N$ matrix of M user reviews and N word attributes. $M_i N_j$ represents the frequency of a word. We employ the term frequency–inverse document frequency (TF-IDF) as a means to increase the weight of words that occur frequently in a single user review and decrease the weight of words that occur frequently in many user reviews [21]. For example, the word 'app' would be penalized because it occurs in many user reviews whereas an uncommon word like 'omission' would be given a higher weight. $f_{i,j}$ is the frequency of a word i in user review j:

$$\textit{TF-IDF}(word_{i,j}) = f_{i,j} * log(\frac{total\_user\_reviews}{total\_user\_reviews\_with\_word\_i}) \quad (3)$$

2.3.2 Multi-labelling Approaches

In this paper, we experiment with several different multi-labelling approaches *e.g.*, Binary Relevance (BR), Classifier Chains (CC), and Pruned Sets with threshold extension (PSt) as well as several different classifiers *e.g.*, support vector machines (SVM), decision tree (J48) and Naive Bayes (NB).

Selection of approaches and classifiers We selected three approaches for multi-labelling the user reviews. The approaches transform the problem of classifying multi-labelled data into one or more problems for single labelling. For example, if a multi-labelled problem had two labels, the approach would predict if the user review contained the first label, the second label, both labels, or neither. Such prediction is accomplished by building two classifiers, a classifier for label one and a classifier for label two. The results from both classifiers are combined. Such a problem transformation allows standard discriminative machine learning models like SVM to be used to multi-label user reviews.

As Figure 5 demonstrates, BR transforms the problem into many single-labelled problems. BR will construct N binary models for N labels. The models can be any binary machine learning algorithm. The main weakness of BR is that it does not leverage the correlations between labels. BR's loss of information is problematic because the issue types in our dataset are correlated (*i.e.*, some of them have high co-occurrence probabilities) as shown in the results of our preliminary study (see Section 2.2.3) *i.e.*, a network problem is more likely to be in a user review that has a bug complaint.

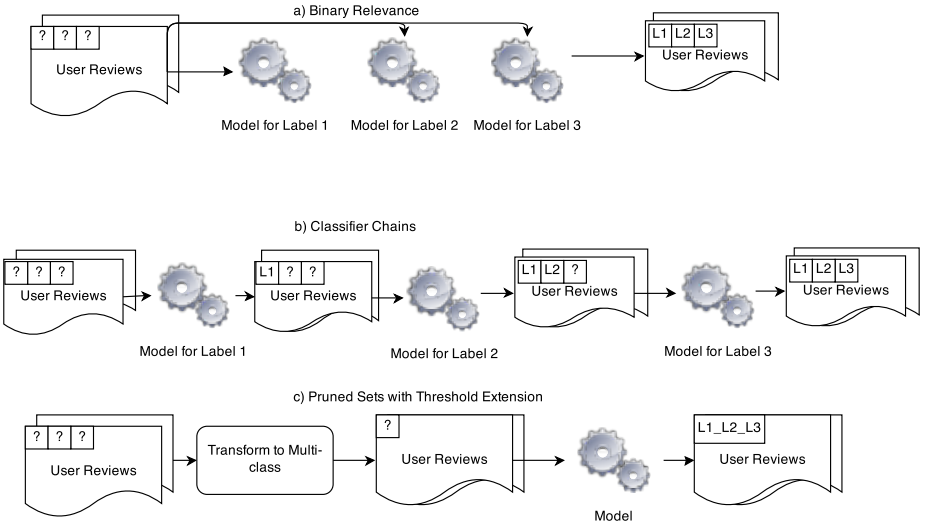

Fig. 5: Examples of Binary Relevance, Classifier Chains and Pruned Sets with Threshold Extension

CC extends BR by building models in a serial fashion and every k^{th} model takes into account the prediction of the $k-1^{th}$ model. As such, CC does not assume independence between labels and correlates labels with one another [22].

PSt differs from BR and CC by treating each possible multi-label combination as a value in a single label. PSt leverages the correlations between labels. PSt produces a large number of possible outcomes that contain each single label *e.g.*, 2^n possible combinations exist for n labels. PSt removes infrequently-occurring label combinations and any user reviews that contained the infrequent label combination are duplicated into multiple single labelled user reviews. For example, an infrequently-occurring combination label {*functional complaint, additional cost*} would be removed and the user reviews i containing the combination would be duplicated into two user reviews i_1 {*functional complaint*} and i_2 {*additional cost*}. Furthermore, any combinations that had never occurred in the training data and therefore would be excluded from the list of possibilities would be given a posterior probability. The posterior probability is the percentage of a label's frequency in the training data. For example {*functional complaint, response time, network problem*} which never occurred in the training data would be assigned a probability based on the probability of each label occurring together *e.g.*, probability of {*functional complaint, response time, network problem*} = probability of {functional complaint} and {response time} and {network problem} occurring in the same user review.

Our selection of the SVM, J48 and NB classifiers is motivated by prior studies which demonstrate the good performance of these classifiers with multi-labelled data [15].

The BR, CC, PSt, SVM, J48 and Naive Bayes classifiers are implemented in MEKA. We use the default values of each classifier in MEKA.

Threshold Optimization: Multi-labelled classification requires a threshold. Surpassing the threshold with a prediction confidence indicates that a user review has such a particular label. There is a corresponding threshold value for each label. More formally, in multi-

labelled classification the output of the model is a matrix of $d \times l$ confidence predictions; where d are the user reviews and l are the labels. For example, the model predicts that the label l_i is in user review d_j because the model's prediction confidence is 0.7 for l_i. 0.7 is higher than the threshold 0.6 for l_i. The confidence predictions are real numbers ranging from 0 to 1. The goal of threshold optimization is to determine at what value a confidence prediction denotes class membership of a label. More information on threshold optimization can be found in [23].

We use the *Proportional Cut* threshold algorithm and determine a separate threshold for each of the 14 labels. The thresholds are set based on the occurrence of label values in the training data. This threshold optimization assumes that the test data will have a similar distribution. This algorithm exhibits similar performance as well as other methods in literature [15].

2.3.3 Evaluation

Cross-validation: To test the accuracy of performing multi-labelled classification on our labelled dataset, we perform a 10-fold cross-validation. The data is split into 10 equal groups (*i.e.*, folds). The first group is selected as the testing data and the other 9 groups are selected as the training data. The process is repeated 10 times with a different group chosen as the testing data for each repetition.

Justification of evaluation measures: We provide six evaluation measures for multi-labelled data. The rationale for providing six different measures is for the following reasons: (i) There are multiple different evaluation measures used in the literature for multi-labelled data [24]. (ii) BR's, CC's and PSt's performance, relative to one another, are affected by the choice of evaluation measure [15]. (iii) Each measure has different levels of strictness and penalizes the errors in the classifier differently. (iv) There are two categories of evaluation measures for multi-label classification, *i.e.*, label-set and label-based. Label-set measures (we refer to label-set measures as review-based measures from this point forward) independently evaluate each user review, whereas label-based measures independently evaluate each label. Accuracy and exact match and F-measure micro are review-based measures, f-measure macro (by label) is an example of a label-based measure. An approach may have superior performance in one measure relative to another measure.

Evaluation measures: The first selected measure is called exact match [22]. Exact match is a very strict evaluation measure. Exact match is defined as follows: let $\mathbf{S}$ be the predicted label values, let $\mathbf{Y}$ be the actual label values. Let r_j be a user review. Let n be the total number of user reviews.

$$f(r_j) = \begin{cases} 1 & \text{if } \mathbf{S_j} = \mathbf{Y_j} \\ 0 & \text{if } \mathbf{S_j} \neq \mathbf{Y_j} \end{cases} \tag{4}$$

$$exact\_match = \frac{1}{n} \sum_{j=1}^{n} f(r_j) \tag{5}$$

For example, if a model predicts {functional complaint, response time, network problem} and the actual labels are {functional complaint, response time, crashing, feature request} the exact match score is 0 because there is at least one label incorrectly labelled.

A less strict alternative is multi-labelled accuracy [22]. The intersection of the predicted labels with the actual labels is divided over the union of the predicted labels with the actual labels. Let $\mathbf{Y_j}$ be the actual labels for user review j, let $\mathbf{S_j}$ be the predicted labels for user review j and n be the number of user reviews.

$$accuracy = \frac{1}{|n|} \sum_{j=1}^{n} \frac{|\mathbf{S_j} \cap \mathbf{Y_j}|}{|\mathbf{S_j} \cup \mathbf{Y_j}|} \tag{6}$$

For example, if a model predicts {functional complaint, response time, network problem} for a review and the actual labels are {functional complaint, response time, crashing, feature request} then the intersection is {functional complaint, response time} and the union is {functional complaint, response time, crashing, feature request, network problem}. The accuracy is $\frac{2}{5}$ for the example review.

F-measure is another common evaluation measure in the multi-labelled literature. It is the harmonic mean of precision and recall. Let $\mathbf{z}$ be a vector of 0/1 values of actual labels and $\hat{\mathbf{z}}$ be the predicted labels.

$$precision(\mathbf{z}, \hat{\mathbf{z}}) = (|\mathbf{z} \cap \hat{\mathbf{z}}|)/|\hat{\mathbf{z}}| \tag{7}$$

$$recall(\mathbf{z}, \hat{\mathbf{z}}) = (|\mathbf{z} \cap \hat{\mathbf{z}}|)/|\mathbf{z}| \tag{8}$$

$$F\,measure(\mathbf{z}, \hat{\mathbf{z}}) = \frac{2 * precision(\mathbf{z}, \hat{\mathbf{z}}) * recall(\mathbf{z}, \hat{\mathbf{z}})}{precision(\mathbf{z}, \hat{\mathbf{z}}) + recall(\mathbf{z}, \hat{\mathbf{z}})} \tag{9}$$

There are different ways to calculate the precision, recall and f-measure for the entire dataset. The method of calculation is dependent on how the $\mathbf{z}$ vector is created. Let N be the number of user reviews and L be the number of labels. In the micro version $\mathbf{z}$ is $L \times N$ data points:

$$\mathbf{z} = [y_j^i, ..., y_L^N] \tag{10}$$

In the micro version there is one precision, one recall and one f-measure for the entire dataset. The f-measure micro is a global f-measure across all reviews. The $\mathbf{z}$ and $\hat{\mathbf{z}}$ are calculated per review and then the value of both are summed up across all the reviews. This summation produces two global values that are used to calculate the precision and recall.

$$F\,measure\,micro(D) = F\,measure(\mathbf{z}, \hat{\mathbf{z}}) \tag{11}$$

The $\mathbf{z}$ vector for macro (by label) is of size N:

$$\mathbf{z} = [y_j^1, ..., y_j^N] \tag{12}$$

Given that there are L labels, there are L precision, recall and f-measures. The f-measure macro (by label) is calculated as the average f-measure for each label. The $\mathbf{z}$ and $\hat{\mathbf{z}}$ are summed per label.

$$F\,measure\,macro\,by\,Label = \frac{1}{|L|} \sum_{j=1}^{L} F\,measure(\mathbf{z_i}, \hat{\mathbf{z_i}}) \tag{13}$$

We evaluate the models using the following measures: exact match, accuracy, micro precision, micro recall, micro f-measure and macro f-measure L (by label). Table 6 demonstrates examples of the different measures.

Table 6: Example of evaluation measures

Two Example Reviews	Labels	L1	L2	L3	L4	L5	L6	L7
Review 1	Actual	✓	✓	✓	✓			
	Predicted	✓	✓			✓		
Review 2	Actual	✓	✓			✓		✓
	Predicted					✓	✓	

Measure	Value
Exact Match	$\frac{1}{2}(0+0) = 0$
Accuracy	$\frac{1}{2}(\frac{2}{5} + \frac{1}{5}) = 0.3$
Precision-Micro	$\frac{3}{5} = 0.6$
Recall-Micro	$\frac{3}{8} = 0.375$
Precision-Macro by Label	$\frac{1}{7}(\frac{1}{2} + \frac{1}{2} + \frac{0}{1} + \frac{0}{1} + \frac{1}{2} + \frac{0}{0} + \frac{0}{1}) = 0.214$
Recall-Macro by Label	$\frac{1}{7}(\frac{1}{2} + \frac{1}{2} + \frac{0}{1} + \frac{0}{1} + \frac{1}{1} + \frac{0}{0} + \frac{0}{1}) = 0.286$

Table 7: Performance of a multi-labelled approach on 14 labels

Evaluation Measure	Pruned Sets with threshold extension - SVM
Precision	65%
Recall	64%
F-measure micro	64%
F-measure Macro (by label)	56%
Accuracy	59%
Exact Match	44%

2.3.4 Results

Results for all 14 labels: As Table 7 shows average performance results, we achieve 59% accuracy, 44% exact match, 65% precision, 64% recall, 64% F-measure micro and 56% F-measure macro (by label). The results are well above 1/15 (0.07%), which is the random chance of guessing that a user review contains one of the 14 labels or no label at all. We find the results to be satisfactory as they compare well with prior results of other multi-labelled classification efforts [15]. We leave the improvement in performance to future work as this is a first step.

Ambiguous issue types: As a precaution, we investigated the performance of each label individually. We remove the multi-labelled Google Play Store and Apple App Store user reviews (the single labelled user reviews allows us to observe individual f-measures and precision-recall curves (PRC) for each label). We follow the data preparation steps in RQ2 and run the SVM classifier that is available in WEKA. The PRC is the area under a graph with recall on the x-axis and precision on the y-axis. The PRC ranges from 0 to 1. A PRC of 1 denotes 100% precision and recall. As Table 8 demonstrates, certain labels perform poorly compared to others. In particular, the response time, uninteresting content, and user interface labels perform poorly.

We conclude that future research may be needed to accurately predict these ambiguous issue types at a higher f-measure. Hence, for now we merge these three ambiguous issue labels into the 'other' issue type. In doing so our average f-measure may improve as the model is no longer burdened with the hard to identify issue types.

The fact that we removed the poor performing labels does not invalidate the results of our other labels. We do not remove the data with poor performing labels. We replace the

Table 8: Performance of a classifier on single labelled user reviews from the Apple App Store and Google Play Store

Issue	F-Measure	Precision-Recall Curve Area
Additional Cost	78	84
Functional Complaint	64	69
Compatibility Issue	64	73
Crashing	89	94
Feature Removal	56	61
Feature Request	66	72
Network Problem	63	64
Other	52	54
Privacy and Ethical Issue	45	48
Resource Heavy	64	61
Response Time	23	27
Unintersting Content	19	18
Update Issue	53	49
User Interface	38	34

labels of the reviews with poor performing labels with the "other" label. The model must predict the "other" label correctly the poor performing reviews. The better accuracy is due to the reduced number of labels. An $n-1$ label problem is easier to predict than an n label problem just as a binary problem is easier for a model to predict than a 14 label problem.

Results for 11 labels: We now perform the multi-label classification on the 11 remaining issue types. The results show that the f-measure macro (by label) measure increases by 56% to up to 63%. The increase demonstrates that the merging of the ambiguous issue types with the 'other' issue improved the performance of individual labels.

We find that the LibSVM classifier performs the best amongst the three different classifiers and that PSt performs the best amongst the multi-label approaches. Table 9 shows the results of the various classifiers using the aforementioned cross-validation process on the Google Play Store. We find that CC does not perform as well as BR in the f-measure evaluation measures. The lower performance of CC is likely due to the low label cardinality of 1.26 and the limited number of cross-label correlations for CC to take advantage of. We perform the Kruskal-Wallis statistical test ($x = 0.05$) on the accuracy evaluation measure to determine if the differences in the results are statistically significant 1) between the three models: Naive Bayes, J48 Decision trees, and LibSVM 2) between the three approaches using LibSVM: BR, CC, and PSt and 3) between all nine multi-labelling model combinations. The differences in performances are statistically significant. The findings hold even after applying a bonferroni correction.

The approaches that we use in this study *i.e.*, BR, CC, and PSt have known biases [15]. The performance of the BR and CC approaches favors label-based evaluation measures *e.g.*, f-measure macro (by label) whereas the performance of the PSt approach favors review-based evaluation measures [15].

Pst and BR have similar precision, recall and f-measure micro results using LibSVM. PSt outperforms BR and CC according to the accuracy measure and exact match measure. However, BR outperforms PSt and CC according to f-measure macro (by label). We ultimately, choose PSt as it has superior accuracy and exact match results compared to BR which has only superior f-measure macro (by label).

Table 9: Performance of approaches on the user reviews from the Apple App Store and Google Play Store

Aproaches	**Binary Relevance**			**Classifier Chains**			**Pruned Sets with threshold extension**		
Classifiers	**Naive Bayes**	**Decision Tree**	**SVM**	**Naive Bayes**	**Decision Tree**	**SVM**	**Naive Bayes**	**Decision Tree**	**SVM**
Precision	17	38	66	17	53	**75**	32	47	66
Recall	76	55	65	**82**	51	54	38	49	65
F-Measure Micro	28	45	**65**	28	52	63	35	48	**65**
F-Measure Macro L	28	48	**63**	27	47	60	31	43	62
Accuracy	21	33	56	20	47	52	27	45	**60**
Exact Match	0.2	11	41	0.17	36	42	11	33	**45**

Our multi-labelled classification of mobile user reviews has a precision of up to 66% and a recall of up to 65%.

3 Applications of our Multi-labelling Approach

As we have shown, we can label user reviews with 66% precision and recall of up to 65%. Therefore, we wish to demonstrate the application of our approach. In other words, how each of the three main stakeholders of mobile apps would benefit from the availability of automatically labelled reviews. For future work, we wish to study the usefulness of our proof-of-concept scenarios for stakeholders. For now, we address the applicability of our approach. We pose the following research question:

RQ3: Are multi-label approaches useful for stakeholders?

To address RQ3, we define three proof-of-concept scenarios: app comparison, app store overview, and anomaly detection. Each scenario requires the analysis of user reviews.

The datasets used in these proof-of-concept scenarios are separate from the reviews downloaded in our manual analysis because the previous reviews were selected from 20 apps and our proof-of-concept scenarios require a much larger number of apps to generalize our results.

We apply a PSt approach with an SVM model because the combination of PSt and SVM exhibits the best performance (see RQ2).

Cross-validation has been shown to not be entirely reliable as an estimate [25]. Therefore, to better understand the performance of our multi-labelling approach on the 601,221 Google Play store user reviews, we randomly select a statistical sample of these reviews with a confidence level of 95% and a confidence interval of 5%. We select 384 user reviews. An undergraduate student verified all the automatically labelled user reviews. The undergraduate student (who is not a co-author) was a volunteer. The results of the undergraduate student were verified by the first author to reduce any bias. If we disagreed we came to a consensus. We disagreed on 7% of the user reviews. We build a model on the previously labelled data and test the model on the random sample. We achieve an accuracy, exact match,

Table 10: Performance of our multi-label approach on a random sample of the Google Play Store user reviews

Evaluation Measure	Pruned Sets with threshold extension - SVM
Precision	50%
Recall	62%
F-measure micro	55%
F-measure Macro (by label)	52%
Accuracy	49%
Exact Match	35%

precision micro, recall micro, f-measure micro, f-measure macro (by label) of 45%, 32%, 46%, 59%, 51%, 50% respectively as Table 10 demonstrates.

The drop in the performance of the model on the random sample compared to the previous performance of the model using cross-validation is expected. There are apps involved that may have specific issues that were not encountered in the training data as well as new apps with new problems. Increasing the size to consider additional apps and reviews in the training data would improve the performance. These steps are left for future work. Additionally, similar drops are reported in papers that contain a training set, a test set, and a validation set [26, 27]. The common practice in machine learning is to train models using the training set and adjust the parameters and types of machine learning techniques to produce a model that performs best on the test set. The model's performance is then tested on a validation set that was not part of the model selection process. We follow this approach by selecting the best model out of a possible nine models that performs best against the 10-fold cross validation. The validation set in our case is the Google Play Store data that we used to apply our three applications on.

The performance of our approach is suitable for the following three proof-of-concept scenarios because for each scenario, we concentrate on spikes in data. We are not concerned with small differences between issues in apps. We draw our results from significant differences in issues in comparison to other apps. Additionally, our approach is a prototype with improvements to performance left for future work.

3.1 Scenario 1 - App Comparison

Motivation: Developers and users benefit from the ability to compare apps. Developers can conduct competitive analysis and users can conduct comparative analysis [2, 3].

Competitive analysis is the comparison of competing apps with similar features and target user base. Competitive analysis is useful for developers. For example, if a developer already has a weather app, competitive analysis would enable him to track the competition, to observe what users are asking for in competing apps. Furthermore, if a mobile app developer wants to develop a new weather app, he could observe the problems faced by similar apps and attempt to avoid them *e.g.*, what features should be added, avoid privacy violations.

Comparative analysis is the comparison of the available information about a product before making a purchase. For example, users may wish to download a weather app, but the app store only offers rudimentary tools to compare the many available weather apps (the store only shows raw ratings and user reviews). If two weather apps have the same rating, *e.g.*, 4.0, a user would turn to the user reviews to help inform their choice. However, a user would be forced to read many user reviews to gain an understanding of the issue types that

users are having. If an app store has graphs of issue distributions then the issue distribution would be helpful for a user to make an informed decision when purchasing/downloading an app *e.g.*, one app is reported by users to crash much more than the other. Graphs of issue distributions are not currently available on app stores.

Data Collection: We collected 6,532 user reviews from three apps from the Google Play Store in the weather category. We preprocess and retain only one-star and two-star user reviews for a total of 536. We use an open source Google Play Store crawler to extract the apps' information and user reviews [28]. Each run collects the 500 newest submitted user reviews. We ran the crawler four times between a period of about three weeks (August 20th 2013 to September 9th 2013) to allow time for the submission of new user reviews.

Approach: We select the three most popular apps from the weather category with similar ratings between 4.1 and 4.4 and over 1 million downloads. We choose the weather category because most of the apps have a similar feature set and purpose *i.e.*, to show the current weather and future weather forecasts. In total, there are 210, 89, and 237 one-star and two-star user reviews for the 'AccuWeather', 'WeatherBug', and 'The Weather Channel' apps respectively. We run our multi-labelling approach on the reviews of these selected apps.

Results: We find spikes in the frequency of occurrences of certain issue types between the apps. Figure 6 gives a more detailed overview of all three weather apps. The spike in 'feature requests' for 'The Weather Channel', compared to the other two apps, is due to the feature request for the ability to turn off the notifications (*e.g.*, "They added a notification to show the current temp. Nice if you like it but you can't turn it off and it stays there when if you exit the app."), which is not present in the other two apps. The spike in 'network problem' for 'AccuWeather' is due to the time that it takes to refresh the information in the app. An example of a review is "This app stopped refreshing and would not upset weather info when refreshing manually". 'AccuWeather' also crashes at a higher frequency (*e.g.*, "i like the App info but beware, it crashes a lot!!!!").

The most important issue types for users, measured by frequency, are different amongst the three apps. Users of 'The Weather Channel' complain about (from most to least): functional complaint, feature request and update issues. For 'WeatherBug', the issue types are update issue, functional complaint and crashing. For 'AccuWeather', the issue types are functional complaint, crashing and network problem. A developer could focus on the issue types most pressing to their users as demonstrated in Figure 6.

There are not many 'resource heavy' issues. There are not many compatibility issues either, the lack of compatibility issues means that the developers might not have to worry as much about doing extensive testing across different phones and different operating system versions [29].

Overall, a user might wish to avoid downloading the 'AccuWeather' app because it has several spikes in 'crashing' and 'network problem' issues compared to the other two weather apps. However, if one were to choose between 'The Weather Channel' and 'WeatherBug', we observe that 'WeatherBug' crashes more but has many less feature requests suggesting that the app is unstable but it might contain most of the features that users want. Our manual analysis of 'The Weather Channel' reviews indicates that most of the feature requests are about the notification bar such as "There is no option to turn off the persistent notification.".

A user may prefer a more stable app to an app that is missing desirable features, while another user may prefer the latest features and accept the instability. The knowledge of the trade-off allows users to make an informed decision.

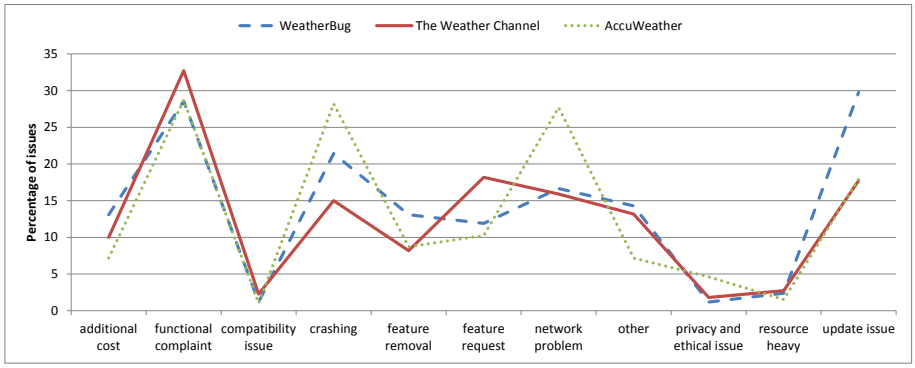

Fig. 6: Comparison of 3 similar rated popular apps with similar ratings in the Weather category

A developer of 'The Weather Channel' app can take action based on the knowledge that the app has more feature request issues than its competitors. Also, the developer of the 'AccuWeather' app could prioritize fixes and stability issues because of the spikes in the labelled network problems and crashing reviews.

3.2 Scenario 2 - App Store Overview

Motivation: Observing the distribution of issue types for an app store is beneficial to the owners of the app stores. Owners would be able to observe the issue types with the most user complaints. Owners could provide support in the form of Frequently Asked Questions (FAQs), tools, guidelines and policy changes in proportion to the issue distributions occurring in their app store. The market owner could investigate why an issue type occurred so frequently relative to other issue types. The issues may be the fault of the app developers but if they occur for many developers, the market owner could take action in the form of software updates and API changes.

Developers would benefit from knowing if the user reviews of their app have differences in the frequency of issue types between two different app stores (*e.g.*, Apple App Store versus Google Play Store). The ability to perform such cross-store comparisons is of great value to developers, since the median number of platforms (stores) that developers develop for is two [8]. For example, if an app on store A has more compatibility issues reported by users than on store B, the app's developer could investigate why their app had more compatibility issues in store A and perform more compatibility testing in the future for store A.

Data Collection: We collected 3.7 million user reviews from 12,000 apps from the Google Play Store. We selected the top 400 apps in the USA across the 30 different categories (*e.g.*, Photography, Sports and Education) based on Distimo's ranking of apps. Distimo is an app analytic company [5]. We again use the crawler mentioned in Section 3.1 to download 500 reviews of the top 400 apps across the 30 different categories of the Google Play Store. We ran the crawler twice, several weeks apart, to allow time for new user reviews to appear. Also some apps did not have 500 new reviews even after the second crawler run. Hence, the

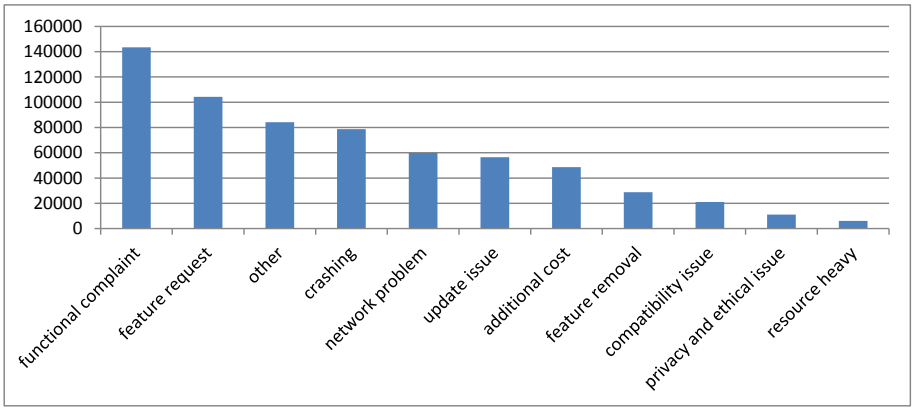

Fig. 7: Distribution of issue types for 1 & 2 star user reviews in the Google Play Store dataset

actual amount of user reviews that we crawled is lower than the expected amount of user reviews *i.e.*, 6 million (500 reviews for each of the 12,000 studied apps).

We follow the same preprocessing steps as RQ2 (Section 2.3.1). From the initial 3.7 million user reviews, our selection of one-star and two-star reviews and preprocessing steps reduce the amount of reviews to 601,221 user reviews.

Approach: We first run our multi-labelling algorithm on the 601,221 user reviews to determine the distribution of issue types across the store. We then separate the data into the 30 categories from the app store *e.g.*, social, finance, and education.

Lastly, to observe if there are differences between the same app across different stores, we compare two apps across both the Apple App Store and the Google Play Store. We select all the one-star and two-star user reviews for the Facebook and Kindle apps in our dataset for a total of 534 Facebook and 355 Kindle user reviews.

Results: Figure 7 shows that the dominant issue types for the Google Play store are functional complaints, feature requests, and network problems. We also observed the issue distributions per category, *e.g.*, business, comics, sports etc. Table 11 displays issue type percentages that are more or less than two standard deviations above the mean for each category in one-star and two-star user reviews in the Google Play Store. We investigated several of the categories. The finance category has higher 'functional complaint' issues than others, *e.g.*, mobile cheque processing does not work, login errors and unprocessed requests. The Shopping category had a higher number of 'feature requests' which included requests for better payment options and improved ease of use. The Shopping category also had higher 'privacy and ethical issues' which included complaints relating to unredeemed coupons or being asked to enter personal details before being able to claim a deal. The Cards category contained higher 'additional cost' issues as many employ a free trial version, known as a freemium model, that may disappoint users [30]. The Brain and Personalization categories contained user reviews that complain about ads and notifications. The Weather and Productivity categories contained user reviews that mention unnecessary background processes that drained the battery. The Tools and 'Libraries and Demo' categories contained higher compatibility issues. The reviews reported lack of compatibility with specific phones and versions of the Android operating system.

Table 11: Store categories that have specific issue types above or below two standard deviations of the mean in our Google Play Store data set

Issue Type	**Two Standard Deviations Above Mean**	**Two Standard Deviations Below Mean**
Functional Complaint	Finance	Racing
Feature Request	Shopping	
Other	Racing	
Crashing	Sports, News and Magazines	
Network Problem		Personalization
Update Issue	News and Magazines	
Additional Cost	Cards	
Feature Removal	Brain, Personalization	
Compatibility Issue	Libraries and Demo, Tools	
Privacy and Ethical Issue	Shopping	
Resource Heavy	Productivity, Weather	

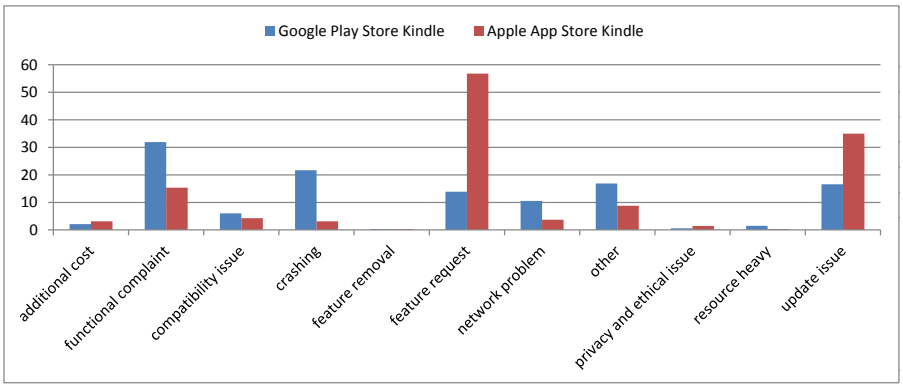

Fig. 8: Comparison of the Kindle app for the Google Play and Apple App Stores

We did not find any major deviations between the Facebook app of the Apple App Store and Google Play Store. However, there is a large spike for feature requests in the Kindle app on the Apple App Store as Figure 8 demonstrates. We investigate the reason and find that due to Apple's store regulations, Amazon, the developer of the Kindle App, had to remove the ability to buy books from within the Kindle app. The removal of this feature was a major issue for many users. The Google Play Store hadn't taken that step, hence the Google Play version of the app didn't have those feature requests.

3.3 Scenario 3 - Anomaly Detection

Motivation: App store owners have specific developer content policies [31] and core app quality guidelines. For example, the Google Play Store has quality criteria for apps [32] that developers must follow. App developers are strongly recommended to follow these guidelines. These policies and guidelines are in place to ensure a certain baseline quality experience for the users. If a developer does not follow the guidelines then an app store may remove his app from the store. This is comparable to how grocery stores set a minimum

quality standard for their products *e.g.*, removing spoiled products (e.g., rotten fruits) from the customer's potential choices. One solution is to read the user reviews in order to find apps that might violate the quality criteria of a store. However, in order for an app store to perform this task, the store needs to analyze and read through millions of reviews across thousands of apps to find troublesome apps. Reading through the user reviews manually is a time consuming task. Therefore, an automated approach to analyze the millions of reviews would be helpful to app store owners.

The app store owner could analyze the issue distribution of all apps within their store and automatically flag apps where one or more issue types occur above a threshold relative to all other apps in the store. The app store owner could warn the developer or remove the offending app. The app store owner could support their decision through evidence gathered from user reviews.

Data Collection: We use the same data collected for scenario 2 in Section 3.2.

Approach: We analyze the issue distribution of our sample of the Google Play Store data set in order to demonstrate the effectiveness of our anomaly detection approach for user reviews. We use control chart theory, which was invented to monitor manufacturing processes and detect deviations [33]. Control chart theory has been applied to software engineering problems in other contexts (see [34, 35, 36]).

We calculate the average and standard deviation for each issue type across all the apps (we exclude apps with less than 15 one-star and two-star user reviews to prevent apps with few reviews from being detected as outliers). If an issue type L_j occurs in the user reviews of an app with a frequency greater than x standard deviations of the average for all apps, we flag the app as anomalous. Traditionally, in control chart theory three standard deviations are used as the threshold for anomaly detection. We perform experiments on one, two and three standard deviations and compare the results. The actual deviation level can be determined by practitioners based on the amount of effort and time that they might wish to spend on this type of analysis.

For example, if app A had 80% of its user reviews labelled as crashing while the average amount of user reviews labelled as crashing was 30% and the standard deviation was 15% for all apps across the store, then app A is above the average + two standard deviations and will be flagged as anomalous.

$$flag(L_j) = \begin{cases} 0 & \text{if } L_j <= 2 \times std\_dev \\ 1 & \text{if } L_j > 2 \times std\_dev \end{cases}$$

Results: Our initial results are displayed in Table 12. The higher the standard deviation, the more issues the apps have of a certain category. There are many more crashing apps than resource heavy apps as flagged by our approach.

We select two standard deviations as our threshold to further analyze the apps. Our choice of two standard deviations is arbitrary but can be adjusted by an analyst based on their needs (*e.g.*, the number of apps they can manually inspect, the severity of each issue type). Our reasoning was that amounts of issues that occurred with 2 standard deviations outside the mean were statistically rare and worth investigating.

Our approach flags 9 apps as anomalous for the 'feature removal' issue type, 36 for 'privacy and ethical issues', 3 for 'resource heavy', and 73 for 'crashing'. Table 13 provides examples of anomalous apps for each issue type. We manually analyzed the reviews of the most anomalous apps for feature removal, privacy and ethical issue, resource heavy and crashing and highlight several example apps. Figure 9 demonstrates the number of apps that

Table 12: Percentage of anomalous apps that are inaccessible in the store (1 year later) and anomalous apps flagged by our approach for 1, 2 and 3 standard deviations. N/A denotes there were no apps at that range

	1-std dev		2-std dev		3-std dev	
Issue Type	Inaccessible	Total Flagged	Inaccessible	Total Flagged	Inaccessible	Total Flagged
Privacy and Ethical Issues	15%	93	13%	31	20%	5
Feature Removal	23%	44	50%	8	100%	1
Resource Heavy	N/A	N/A	N/A	N/A	33%	3
Crashing	9%	192	15%	61	0%	12

Table 13: Most anomalous apps from the four categories that were flagged in our analysis of the Google Play Store dataset

App Name	**Issue**	**Comments**
ESPN Bracket Bound 2013	Crashing	"Constantly crashes and won't let me look at my brackets. Thanks for putting out another crappy app ESPN" "This app crashes and has to restart literally every time I open it. This is a worthless app. Sucks" "Crashes about 10 times a day. Extremely annoying" "This would be a great app if it didn't crash every time I use it"
Pic Stitch	Feature Removal	"It sends ads as notifications to my phone. And as others stated...too many in app adds. Just downloaded last night and am now deleting." "1 minute in and my notifications is full of ads.... NO THANK YOU" "THERE IS WAY TOO MUCH SPAM AND ADS TIED TO THIS APP." "Ads all over the place AND spam notifications on my phone...DELETE!!"
Cabela's	Resource Heavy	"This app will run in your background and absolutely destroy your battery life" "This app uses the gps to know your location. It exhausted up my battery in less than half a day. After removal of the battery usage was fine again" "nice when shopping but eats up battery quick and you can't stop it" "Used 31% of my battery without even opening the app. Not worth having in case I catch a tagged fish!"
Quote Rocket Insurance	Privacy and Ethical Issue	"This app is a piece of crap they spam you all the time and telemarketers call you all the time and it steals your information its called Quote Rocket it didn't give my quote this stupid app wastes your time it is just a horrible piece of crap don't get it I am warning you" "Too much information needed to give out." "It just steals your information. You end up getting calls from stupid telemarketers." "I go through everything & don't get my crystals for my game & you can't help me with Insurance for my car" "I did everything and filled it out correctly. Got an email with my quote and a phone call still no coins. :("

exceed our threshold. We note that an app store owner may define their own thresholds and it is entirely dependent on how sensitive the app store owners are to each issue. We investigate the user reviews of apps and find reports of direct violations of the policies and guidelines of the Google Play Store.

The 9 apps that were anomalous for feature removal issues had user reviews that contained many issues about ads. Users specifically mentioned frequent and obtrusive pop-up ads and ads in the notification bar. The app 'Pic Stitch' had many reports of ads in the notification bar and adding shortcuts and programs unknowingly to the user's phone. This reported behavior is not permitted under Google's developer policy as it states "Apps and their ads must not add homescreen shortcuts, browser bookmarks, or icons on the user's de-

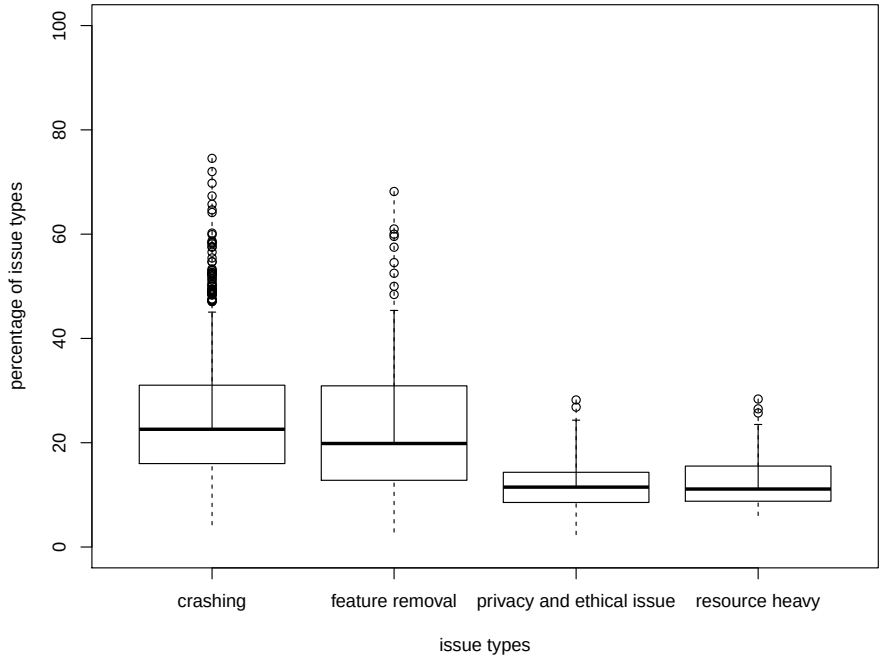

Fig. 9: Box-plots with the anomalous apps as points in the graph above two standard deviations of the mean

vice as a service to third parties or for advertising purposes." [31] Additionally, "Apps and their ads must not display advertisements through system level notifications on the user's device." [31]. The app 'Pic Stitch' could be removed based on its violation of Google's policies if true. In fact, it no longer exists in the Google Play Store as of July 2014.

The app 'Quote Rocket Insurance' flagged as anomalous for privacy and ethical issues stated that if the user downloads the app and enters their contact information (for telemarketing purposes) then the user would be rewarded with an in-game currency for a videogame app. However, user reviews contained reports that users did not receive the in-game currency and that they were still contacted by telemarketers. The second app advertised that users would be given a $20 gift card for free if they entered their information. Users reported in their reviews that the gift cards turned out to be for two very expensive women's clothing stores and that they were later targeted by telemarketers. Google has a policy that apps cannot exist "where the primary functionality is to drive affiliate traffic to a website" [31]. Additionally "forcing the user to click on ads or submit personal information for advertising purposes in order to fully use an app is prohibited."[31].

'Cabelas', the anomalous app for the resource heavy issues had issues about the battery life. Users reported that the app drained their battery while running as a background process, even when not in use. Google states in their core app quality guidelines that apps should not run background processes unless necessary to the function of the app.

Finally the user reviews on the anomalous apps associated with the crashing issues contained issues about crashing and freezing. Crashing and freezing are basic criteria of the core app quality guidelines.

In July 2014 (one year after our initial crawl) we test if the apps are still available in the Google Play Store by accessing the webpage of each of the anomalous apps. If an app is no longer accessible (i.e. removed from the store possibly due to the flagged issues) through its webpage we mark the app as inaccessible. We find that the percentage of inaccessible apps increases or remains the same for apps at higher standard deviations for three out of the four amomalous issue types. Table 12 demonstrates the percentage of inaccessible apps for four issue types that were one, two and three standard deviations away from the mean for that issue type. In the table, if an app is anomalous at three standard deviations, it will not be part of the apps at one or two standard deviations. Likewise, an app at two standard deviations will not be present at one standard deviations.

We demonstrate that with our approach, app store owners can detect anomalous apps and choose to pro-actively warn developers or remove such anomalous apps from their app store.

We demonstrate through three proof-of-concept scenarios that our multi-labelling approach assists all three stakeholders of mobile apps in the following three proof-of-concept scenarios: app comparison, market overview and anomaly detection.

4 Discussion

In this section, we justify our decision to combine the Apple App Store and Google Play Store user reviews into one dataset. We also discuss the implications of the number of labels to predict in our approach. We investigate poor performing issue types, and present alternative approaches.

4.1 Combining Apple App Store and Google Play Store Data

In our multi-label approach, we wish to have as much labelled data as possible. However, the labelled data came from 2 different stores since the data was from a previous study and it wasn't expected that the data is to be used as training data for our study. Our solution was to combine the data and assume that they could be used as a single training dataset. However, we need to ensure that the two datasets did not have major differences. We judge their similarity by testing their respective prediction power on one another.

We randomly selected 1,066 user reviews from Apple App Store and an equal amount of Google Play Store user reviews from our manually labelled data (see Section 2.2.2). We built a model (we followed the approach in Section 2.3.1) on only the Apple App Store data and tested our model on the Google Play Store and vice versa.

As Table 14 shows, the prediction power of both models remains the same. Thus, we conclude that the user reviews are similar enough to be combined into a single data set.

4.2 Impact of Reducing the Number of Labels

We show in our previous single-labelled classification (see Section 2.3.4 where we can see individual f-measures) that the classification of certain issue types perform poorly compared

Table 14: Comparison of training on Google Play Store and testing on Apple App Store and vice versa

	Train	**Test**	**Train**	**Test**
	Google Play Store	**Apple App Store**	**Apple App Store**	**Google Play Store**
Accuracy	34		34	
Exact Match	20		20	
Precision	40		44	
Recall	43		40	
F-1 Micro	41		42	
F-1 Macro L	27		30	

to others. We also see from merging these issue types into the 'other' issue that the overall f-measure improves. Merging issue types demonstrates that as you decrease the number of labels the performance is increased and if you remove the bad performing labels the performance will increase even more.

Iacob & Harrison have proposed approaches to extract feature requests from the user reviews of mobile apps [16]. Their proposed approach used a binary classifier. In other words, they only cared if a review was or was not a feature request. The proposed approach had 96% precision and 86% recall respectively. Their results support our aforementioned observation that a smaller number of labels lead to improved classification performance.

The choice of issue types should be decided based on the interest of each stakeholder and the specific analysis to ensure the best performing automated classification.

4.3 Poor performing issue types

We investigate the reason behind the poor automated classification performance of some of the issue types. As Table 8 shows, certain issue types are more difficult to automatically classify. In particular, response time, uninteresting content, and user interface performed below a 50% f-measure.

The goal of this subsection is not to improve the classification accuracy but instead to understand the reason for the low performance of some of the labels. Hence, we use Labelled Latent Dirichlet allocation (LLDA) which showed that these labels are ambiguous (i.e., have a common set of words).

LLDA is a supervised learning version of the unsupervised approach *Latent Dirichlet Allocation* (LDA) [37]. LDA is a popular statistical topic model [38]. It generates topics from the corpus of documents and then represents the documents as a set of topics with associated probabilities. A topic is a mixture of terms drawn from the documents with associated probabilities for that topic. To generate the topic and term distributions, a generative process is applied to learn the correct distributions such as Gibbs Sampling or Collapsed Variation Bayes. A weakness of LDA is that the number of topics must be defined beforehand as LDA makes no assumptions on the number of topics per document.

LLDA differs from LDA in that it requires documents to be labelled. LLDA constrains the topics to only the labels assigned to the training documents. Hence, the labelled topics correspond directly to the issue types. For example, LLDA will generate a list of words that are most associated with each issue type and infer which documents are most associated with each issue type. LLDA can handle multi-labelled problems inherently [37]. The output of LLDA is a probability distribution for each label and for each document. The output

of LLDA can be used as a model to predict the label of user reviews. In order to make predictions a threshold value is required to be optimized.

We build an LLDA model with the Stanford Topic Modeling Tool[1]. We use the same settings as [39]. The topic and term smoothing parameters are set at 0.1 and 0.1 respectively. The number of sampling iterations are 1,500. Document similarity is computed using the cosine similarity measure. To infer the distributions of words and topics, we use collapsed variational bayes (CVB) inference.

We investigate the three ambiguous issue types *e.g.*, response time, uninteresting content, user interface, that we removed because of their poor performance for automated classification. We find that the most occurring words in response time are similar to network problem *e.g.*, 'slow','takes','very'. We also find that user interface and uninteresting content share similar words with functional complaint and feature request *e.g.*, 'out','please','like'.

The small number of manually labelled user reviews for a particular issue type didn't factor into the classification performance of individual issue types. The resource heavy issue had comparably little training data (just 0.4 and 6 percent of the Apple App Store and Google Play Store data respectively) yet its f-measure was near the average. We find the most probable word associated with the resource heavy issue is the word 'battery' which did not appear in any of the lists of most probable words associated with other issue types.

4.4 Alternative approaches

In this section we present and discuss unsupervised alternatives to our multi-labelling approach for labelling reviews and for performing anomaly detection. We first present an overview of supervised learning vs. unsupervised learning, followed by the results of the unsupervised alternatives.

Our approach uses supervised learning. Supervised learning builds a model that learns the correlations between training examples and the example's labels to predict future example's labels. An unsupervised approach is an alternative to a supervised approach. There are no pre-specified labels. An unsupervised model infers correlations between the data without any labelled training data. The benefit of an unsupervised approach is that it does not require any labelling (a potentially time-consuming or otherwise impossible task). The drawback of unsupervised learning is that the model may infer connections that otherwise are unimportant or unwanted. Hence, is a trade-off between supervised and unsupervised learning depending on the goal.

4.4.1 Unsupervised labelling

We first present an unsupervised approach to our multi-labelling approach. The approach we use is topic modelling. Topic modelling associates topics to documents based on words in the documents. This association is done in an unsupervised (i.e., discovery/exploration-oriented fashion). As such, the topics are related by frequency and commonality instead of the topics of interest for a particular research problem. In our case, we had a specific research problem at hand (the concept of quality of an app). Hence, we did not consider topic modelling, instead we chose a supervised approach. A manual process was followed to identify issues and topics with our particular research problem in mind. The identified

[1] http://nlp.stanford.edu/software/tmt/tmt-0.4/

Table 15: Examples of general topics about photo and videogames with LDA (14 topics) and specific topics about gps and requesting help with LDA (150 topics)

Example topics in LDA			
topic size: 14		topic size: 150	
1	photo, app, picture, better, edit, adobe, use, good, upload, pic, does, photoshop, quality, feature, like, option, thing, useless, image, need	1	list, shop, search, recipe, result, disappear, click, button, view, great, title, ingredient, does, icon, feature, box, bug, groceries, menu, differ
2	game, play, video, watch, movie, good, need, like, great, money, sync, just, upgrade, time, star, coin, farmville, fun, really, spend	2	locate, await, place, mile, area, say, show, find, far, map, nearby, right, accuracy, check, state, try, close, horrible, gp, tip

Table 16: Results for keyword search

Label	Search Terms	Accuracy Percentage
Network problem	network	0
Compatibility issue	compatibility	0.65
Resource heavy	resource, heavy	4.54
Update issue	update	62.9
Bug fix	bug	6.81
Additional cost	addtional, cost	7.1
Privacy and ethical issue	privacy, ethical	3.1
Feature request	feature, request	9.5
Feature removal	remove	4.3
Crashing	crash, crashing	79.9

problem-relevant topics were then fed into an automated classifier in order to locate other occurrences of these topics.

To study the use of unsupervised learning, we perform a new experiment. In the new experiment, we ran topic modelling (i.e., LDA) on our manually labelled data in order to uncover unsupervised topics. We did two runs with the number of topics (K) equal to 14 and 150 topics. We chose 14 since that is the number of topics that were uncovered by our manual supervised approach and we chose 150 topics to get a feel of how well LDA would perform on a much large number of topics [40].

The topics recovered using K=150 are fine-grained app-specific topics instead of being topics that talk about quality-related concerns (the focus of our research). Such examples included topics on GPS or shopping lists as Table 15 shows. The topics created by the 14 topic run are too general to be useful. For example, one of the topics is about media *e.g.*, game, play, video, watch and movie and another topic is about photos *e.g.*, photo, app, picture, better and edit. In Table 15 we see two examples of such topics.

The second unsupervised approach we perform is keyword search. Keyword search is very simple. The approach searches for related words of a label in the review and if the search words match, the review is considered a part of the label. The drawbacks of this approach is that it requires extensive knowledge of how users report issues and an exhaustive list of potential search terms. We select the label words as keywords and search within our already labelled training data. If the label words are contained in a review then the approach predicts the review to contain the label. As Table 16 shows, most labels perform poorly with the exception of ‘crashing’ and ‘update issue’. Reviews with ‘crashing’ and ‘update issue’ are likely to contain words like ‘crashing’ or ‘update’ in the review itself. ‘Privacy and ethical issues’ contain more varied language to describe an issue.

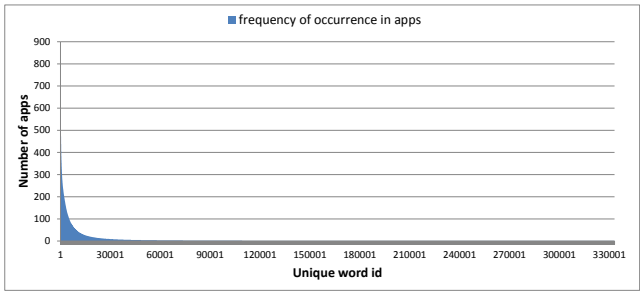

Fig. 10: Distribution of the frequency of occurrence of anomalous words across all apps (sorted by frequency).

Table 17: Top 20 most frequent anomalous (3 standard deviations) words across apps

Words as they appeared in reviews
isnt, thus, means, wouldnt, couldnt wonder, awhile, mostly, provide, deserve, besides, ty, recomend, pleased, opinion, havent, putting, serious, appears, plain

4.4.2 Unsupervised anomaly detection

An unsupervised alternative to detecting anomalous apps with user reviews is a simple bag of words approach. To study the applicability of such a simple approach, we perform the following experiment. We find the average occurrence of every word that occurred in any of the studied reviews. Next, find the 1,2 and 3 standard deviation of each word occurrence. We then flag words that occur outside x standard deviations. We find that the words occurring are not quality-related words. Instead they are simply words that appear in an app because they are specific to an app such as "asl" for a sign language app. The top 20 words are shown in Table 17. Furthermore, we find that the majority of anomalous words do not occur across all apps and are focused in a small selection of apps as Figure 10 shows. The limited coverage of the anomalous words would not be useful for our purposes of differentiating the majority of apps.

5 Threats to Validity

Some threats could potentially limit the validity of our experiments. We now discuss potential threats and how we control or mitigate them.

5.1 Threats to Construct Validity

Since, we manually labelled our gold dataset of reviews with the different issue types, some reviews may have been incorrectly labelled. To mitigate this threat, we performed this labelling in an iterative manner and went over each review multiple times to ensure correct labelling of the reviews. To avoid any bias in the labelling process, one PhD student and a post-doctoral fellow, who are not co-authors, verified the labelling again. If they disagreed on a label they took the majority opinion, *i.e.*, in total three votes, one from the person who

labelled the data and two from persons who verified the data. If the vote was a three way tie they came to a consensus.

The labelled dataset was originally labelled for another purpose [41]. The study of multi-labelled data in user reviews was not the original intention therefore there was no conflict of interest when the labelling occurred to try and show multiple labels.

There was a different amount of Google Play Store and Apple App Store reviews in our training data. This could result in a different number of labels. However, our approach is independent of the number of labels.

5.2 Threats to Internal Validity

We have several parameters that need to be set in our study. The parameters for the SVM tool were chosen based on the default values that experts in the field had determined [42]. We elected to follow their recommendations. The selection of thresholds in our multi-labelling approach could directly impact our results. To mitigate this threat, we selected the proportional cut threshold algorithm to automatically determine the threshold. Proportional cut thresholds for a classifier are determined automatically using the label distributions in the training data. There are other threshold techniques however proportional cut algorithm is shown to be efficient and performs as well as other more complex threshold algorithms [15].

The selection of one-star and two-star ratings could bias the results. We assumed that one-star and two-star ratings contain negative issues. To mitigate this threat, we performed sentiment analysis to confirm our assumptions. The results of sentiment analysis showed that in majority of the cases, one-star and two-star star rated reviews are negative. We recognize that three, four and five star reviews may contain an issue, *e.g.*, feature requests, in an overall positive review. We accept that we could have missed some more issues by eliminating all the three, four, and five star ratings. However, our multi-labelling approach is rating independent. Thus, adding more reviews and/or labels could produce similar results. However, more experiments are needed to generalize our findings on all the star ratings. Also lower ratings are of a much greater concern to developers since they decrease the overall rating of their app in the app store and addressing the concerns of those users is of a greater priority to developers than three, four and five star reviews [41].

The selection of multi-labelling approach, *i.e.*, PSt, could directly impact our findings. Our choice of PSt is based on its performance which we found to be superior to BR and CC.

5.3 Threats to External Validity

Our training set of labelled reviews data is composed of 20 mobile applications. Hence our results may not generalize to all mobile applications. To mitigate this threat, we selected apps from a diverse set of categories, by selecting apps from high and low-rated apps and from the 2 most popular app stores. We employed a random sampling process for each app to collect a representative sample with 95% confidence and 5% confidence interval. The exact number of collected user reviews varies between apps as the total number of user reviews varies per app.

To make our findings more generalizable, we attempted to collect a representative sample of the negative user reviews from the Google Play Store. Therefore, we chose to download from the top 400 most popular apps from Distimo in 30 different categories. Distimo determines popularity by download count and ratings. We acknowledge that this is not a

statistical sample but we required apps that are reviewed regularly and there was no feasible way to generate a list of all Google Play Store apps from which we could select a random sample.

6 Related Work

In this section, we survey the work related to our study. First we discuss some of the previous work related to mobile apps and their reviews. Then we discuss some of the previous work which used automated approaches to label software engineering data.

6.1 Work Related to Mobile User Reviews

Previous work has confirmed that reviews of mobile apps have a major impact on the success of an app [43, 44, 45]. User reviews contain information that could help developers improve the quality of their apps, and increase their revenue. Kim *et al.* [45] conducted interviews of app buyers and discovered that reviews are one of the key determinants in the user's purchase of an app. Similarly, Mudambi *et al.* [44] showed user reviews have a major impact on the sales of online products. Harman *et al.* [43] have shown a strong correlation between app ratings and the total downloads of an app.

Khalid *et al.* manually analyzed and categorized one-star and two-star mobile app reviews [13]. They manually identified the different issue types in mobile app user reviews. They did not automatically label reviews nor did they identify the multi-labelled nature of the reviews. We use their data as an oracle.

Fu *et al.* performed topic modelling on mobile app user reviews [46]. They identified reviews that did not match the given rating, presented an approach to study the evolution of an app's reviews and compared different categories of apps in terms of the 10 topics that they discovered using topic modelling. However, our approach is more concerned with providing context to individual user reviews. Topic modelling may miss context in a topic *i.e.*, a review that refers to the word 'feature' may be negatively referring to the removal of a feature versus adding a feature. Also, topic modelling may confuse issue types such as when we found that 'network problem' and 'response time' share similar words (see Section 4.3). Pagano & Maalej manually labelled and analyzed mobile user reviews across all five star rating levels [47]. They were concerned with questions such as how a labelled issue type affects a user's rating, how issue types affect the helpfulness of a review and how the helpfulness affects the length of a review. Our work complements their work as both works manually label user reviews. However, we chose to focus only on the negative complaints of users since 4 and 5 star ratings are dominated by praise and helpfulness. Therefore, we analyzed the distribution of issues that occur in one-star and two-star reviews only. Also we only concern ourselves with complaints of users and not non-complaint oriented tags (*e.g.*, praise, helpfulness, mentions other app) as included by Pagano & Maalej.

6.2 Work Related to Automated labelling

Automated labelling approaches are widely used for sentiment analysis, spam detection and language detection. Melville *et al.* used Logistical Regression (LR) on text within blogs to identify the sentiment of writers [48]. Jindal *et al.* used a LR model to find spam within

reviews of products [49]. Our work focuses on studying user reviews to help stakeholders efficiently identify the various types of issues from their reviews of their apps.

Several researchers have explored the effectiveness of multi-labelling in software engineering [50, 51]. Han *et al.* [50] used LDA and LLDA to understand Android fragmentation identifying vendor-specific bugs of mobile device manufacturers. Ahsan *et al.* [51] demonstrated SVM to be a very effective classifier for labelling multi-labelled software change requests. Ramage *et al.* [37] showed that SVM and LLDA performed similarly on multi-labelled data. The contents of bug reports and change requests are assumed to contain a bug report or change request whereas the contents of user reviews is open-ended and no assumptions can be made as to what a user review will contain.

Some researchers have reported that even developers incorrectly label bug reports and feature requests [52, 53]. To label bug reports automatically, Antoniol *et al.* proposed an approach to label change requests as a bug or a feature request [53]. However, their approach can only handle binary labelled data. In addition, they analyze change requests only. User reviews are different from bug reports and change requests because bug reports and change requests have more structured style and are often much longer. There is an ability to exchange messages with the reporter, the reporter may assign priority and type *i.e.*, whether it is an enhancement or defect report. Also the report can be filed into categories such as 'accepted', 'closed' and 'duplicate'. There are no such mechanisms in user reviews on app stores.

Carreño *et al.* [40] used opinion mining approaches and topic modeling to extract requirements from user reviews. The authors use unsupervised approaches to cluster the reviews into similar requirements. However, in a large dataset such as ours, consisting of millions of user reviews, the reviews will have far more variable wording for each issue which may necessitate the creation of many topics. A practitioner would have to manually analyze the larger number of generated topics in order to find all the requirements and remove duplicates.

Iacob & Harrison [16] built a rule-based automated tool to extract feature requests from user reviews of mobile apps – their approach identifies whether a user review contains a feature request or not. Our approach expands the number of issues to 13 as we are concerned with multiple issue types. Iacob & Harrison's approach would require new linguistic rules for each new issue type which may be difficult to write given the amount of variable ways that users report issues in our large dataset.

Saha *et al.* labelled Stack Overflow questions with a set of 83 word tags. They built an SVM model for each tag, similar to a BR model in multi-labelling. Linares-Vasquez & Poshyvanyk analyzed the issues developers had for mobile development on Stack Overflow. Linares-Vasquez & Poshyvanyk used LDA to discover the topics that were discussed most frequently.

6.3 Work Related to Opinion Mining

Opinion mining is the study of user's sentiment. Opinion mining focuses on discovering the positive, negative or neutral opinion expressed by a user in a review [54]. There are different levels of analysis: document level, sentence level and aspect (word) level [55]. Document level analysis assumes one sentiment [56]. Aspect level opinion mining concerns the pairing of an opinion word such as "like" with an aspect such as "smartphone".

Ganesan *et al.* [57] proposed an approach to automatically summarize users opinions. Their approach generates a summary of users' opinions; independent of different types of opinions in a cluster.

Pak & Paroubek used Twitter as a corpus of data [58] which is similar in many respects to the style of user reviews.

7 Conclusion

User reviews are an important indication of quality of an app. Labelling user reviews is beneficial to stakeholders. However, user reviews are difficult to label considering the unstructured noisy multi-labelled nature of the data. We demonstrate that even with such difficulties, we can effectively and automatically label the reviews to address real world problems of stakeholders. Our work is another step in moving towards leveraging user reviews to improve the quality of mobile apps.

For future work we would like to improve the prediction performance for our multi-labelling approach especially for the ambiguous issue types. We would also like to perform user surveys of the major stakeholders to better understand how they would use the applications that we demonstrated in this paper.

References

1. M. Butler, "Android: Changing the mobile landscape," *Pervasive Computing, IEEE*, vol. 10, no. 1, pp. 4–7, 2011.
2. Flurry. (2014, May) Flurry. [Online]. Available: http://www.flurry.com/solutions/analytics
3. A. Annie. (2014, May) App annie. [Online]. Available: http://www.appannie.com/app-store-analytics/
4. K. Bostic. (2013, Jul.) Google play takes top spot in downloads, but apple's app store still tops revenue. [Online]. Available: http://appleinsider.com/articles/13/07/31/google-play-takes-top-spot-in-downloads-but-apples-app-store-still-tops-revenue
5. Distimo. (2013, Sep.) Google play store, united states, top overall, free, week 35 2013. [Online]. Available: http://www.distimo.com/leaderboards/google-play-store/united-states/top-overall/free
6. Adobe. (2014, May) Mobile analytics. [Online]. Available: http://www.adobe.com/ca/solutions/digital-analytics/mobile-analytics.html
7. Google. (2014, May) Google analytics. [Online]. Available: http://www.google.ca/analytics/mobile/
8. V. mobile, "Developer Economics Q1 2014: State of the Developer Nation," Tech. Rep., 05 2014.
9. D. Pagano and B. Bruegge, "User involvement in software evolution practice: a case study," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 953–962.
10. A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

11. B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 271.
12. M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *J. Am. Soc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1002/asi.21662
13. H. Khalid, "On identifying user complaints of ios apps," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
14. G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.
15. J. Read, "Scalable multi-label classification," Ph.D. dissertation, University of Waikato, 2010.
16. C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the Tenth International Workshop on Mining Software Repositories*. IEEE Press, 2013, pp. 41–44.
17. M. F. Porter, "Snowball: A language for stemming algorithms."
18. ——, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.
19. J. Read. (2013, Sep.) Meka: A multi-label extension to weka. [Online]. Available: http://meka.sourceforge.net/
20. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278
21. A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2012.
22. J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 254–269.
23. R.-E. Fan and C.-J. Lin, "A study on threshold selection for multi-label classification," *Department of Computer Science, National Taiwan University*, 2007.
24. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*. Springer, 2010, pp. 667–685.
25. K. H. Esbensen, D. Guyot, F. Westad, and L. P. Houmoller, *Multivariate data analysis: in practice: an introduction to multivariate data analysis and experimental design*. Multivariate Data Analysis, 2002.
26. L. Michielan, L. Terfloth, J. Gasteiger, and S. Moro, "Comparison of multilabel and single-label classification applied to the prediction of the isoform specificity of cytochrome p450 substrates," *Journal of Chemical Information and Modeling*, vol. 49, no. 11, pp. 2588–2605, 2009, pMID: 19883102. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ci900299a
27. M. Brameier and W. Banzhaf, "A comparison of linear genetic programming and neural networks in medical data mining," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 1, pp. 17–26, Feb 2001.
28. Akdeniz. (2013, Sep.) Google play crawler. [Online]. Available: https://github.com/Akdeniz/google-play-crawler
29. H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan, "Prioritizing the devices to test your app on: A case study of android game apps," in *22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014)*. ACM, 2014.

30. M. F. Niculescu and D. J. Wu, "When should software firms commercialize new products via freemium business models," *Under Review*, 2011.
31. Google. (2013, Sep.) Google play developer program policies. [Online]. Available: https://play.google.com/about/developer-content-policy.html
32. ——. (2013, Sep.) Core app quality guidelines. [Online]. Available: http://developer.android.com/distribute/googleplay/quality/core.html
33. W. A. Shewhart, *Economic control of quality of manufactured product*. ASQ Quality Press, 1931, vol. 509.
34. T. H. Nguyen, B. Adams, Z. M. Jiang, A. E. Hassan, M. Nasser, and P. Flora, "Automated detection of performance regressions using statistical process control techniques," in *Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering*. ACM, 2012, pp. 299–310.
35. S. Ghaith, M. Wang, P. Perry, and J. Murphy, "Profile-based, load-independent anomaly detection and analysis in performance regression testing of software systems," in *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*. IEEE, 2013, pp. 379–383.
36. A. M. Abubakar and D. N. Jawawi, "A study on code peer review process monitoring using statistical process control," in *e-Proceeding of Software Engineering Postgraduates Workshop (SEPoW)*, 2013, p. 136.
37. D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 248–256.
38. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.
39. D. Ramage and E. Rosen, "Stanford topic modeling toolbox," 2011.
40. L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 582–591.
41. H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile app users complain about? a study on free ios apps," pp. 1–1, 2014.
42. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
43. M. Harman, Y. Jia, and Y. Z. Test, "App store mining and analysis: Msr for app stores," in *Proceedings of the 9th Working Conference on Mining Software Repositories (MSR '12)*, Zurich, Switzerland, 2-3 June 2012.
44. S. M. Mudambi and D. Schuff, "What makes a helpful online review? a study of customer reviews on amazon.com," *MIS Quarterly*, vol. 34, no. 1, pp. 185–200, 2010.
45. H.-W. Kim, H. L. Lee, and J. E. Son, "An exploratory study on the determinants of smartphone app purchase," in *The 11th International DSI and the 16th APDSI Joint Meeting*, Taipei, Taiwan, July 2011.
46. B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1276–1284. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488202

47. D. Pagano and W. Maalej, in *Proceedings of the 21st. IEEE International Requirements Engineering Conference*. IEEE, 2013. [Online]. Available: http://mobis.informatik.uni-hamburg.de/wp-content/uploads/2013/07/RE2013PaganoMaalej.pdf
48. P. Melville, W. Gryc, and R. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1275–1284.
49. N. Jindal and B. Liu, "Review spam detection," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1189–1190.
50. D. Han, C. Zhang, X. Fan, A. Hindle, K. Wong, and E. Stroulia, "Understanding android fragmentation with topic analysis of vendor-specific bugs," in *Reverse Engineering (WCRE), 2012 19th Working Conference on*. IEEE, 2012, pp. 83–92.
51. S. N. Ahsan, J. Ferzund, and F. Wotawa, "Automatic classification of software change request using multi-label machine learning methods," in *Software Engineering Workshop (SEW), 2009 33rd Annual IEEE*. IEEE, 2009, pp. 79–86.
52. K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 392–401.
53. G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, ser. CASCON '08. ACM, 2008, pp. 23:304–23:318.
54. 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.
55. M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 168–177.
56. B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 79–86.
57. K. Ganesan, C. Zhai, and E. Viegas, "Micropinion generation: an unsupervised approach to generating ultra-concise summaries of opinions," in *Proceedings of the 21st international conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 869–878.
58. A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining." in *LREC*, 2010.