Automatic Classification of Non-Functional Requirements from Augmented App User Reviews

Mengmeng Lu
State Key Lab of Software Engineering
School of Computer Science, Wuhan University
Luojiasha 430072
Wuhan, China
lumengmeng@whu.edu.cn

Peng Liang*
State Key Lab of Software Engineering
School of Computer Science, Wuhan University
Luojiasha 430072
Wuhan, China
liangp@whu.edu.cn

ABSTRACT

Context: The leading App distribution platforms, Apple App Store, Google Play, and Windows Phone Store, have over 4 million Apps. Research shows that user reviews contain abundant useful information which may help developers to improve their Apps. Extracting and considering Non-Functional Requirements (NFRs), which describe a set of quality attributes wanted for an App and are hidden in user reviews, can help developers to deliver a product which meets users' expectations. **Objective:** Developers need to be aware of the NFRs from massive user reviews during software maintenance and evolution. Automatic user reviews classification based on an NFR standard provides a feasible way to achieve this goal. **Method:** In this paper, user reviews were automatically classified into four types of NFRs (reliability, usability, portability, and performance), Functional Requirements (FRs), and Others. We combined four classification techniques BoW, TF-IDF, CHI^2, and AUR-BoW (proposed in this work) with three machine learning algorithms Naive Bayes, J48, and Bagging to classify user reviews. We conducted experiments to compare the F-measures of the classification results through all the combinations of the techniques and algorithms. **Results:** We found that the combination of AUR-BoW with Bagging achieves the best result (a precision of 71.4%, a recall of 72.3%, and an F-measure of 71.8%) among all the combinations. **Conclusion:** Our finding shows that augmented user reviews can lead to better classification results, and the machine learning algorithm Bagging is more suitable for NFRs classification from user reviews than Naïve Bayes and J48.

CCS CONCEPTS

• **Software and its engineering** → Requirements analysis; • **Machine learning** → Supervised learning by classification

* Corresponding author

EASE'17, June 15-16, 2017, Karlskrona, Sweden

ACM ISBN 978-1-4503-4804-1/17/06…$15.00
http://dx.doi.org/10.1145/3084226.3084241

KEYWORDS

Non-Functional Requirements, User Reviews, Automatic Classification, Textual Semantics

1 INTRODUCTION

Mobile Apps are developed specially for mobile devices such as tablets and smartphones. The leading App distribution platforms, Apple App Store, Google Play, and Windows Phone Store had over 4 million Apps as of June 2016 [31]. The number of downloads is around 1 billion per month in Apple App Store [2]. Users can evaluate an App by giving a rating with a text feedback after they download and use the App. As a type of collective knowledge [14], user reviews contain valuable information which may help developers to better understand user needs and complaints during software maintenance and evolution [2][3]. Making use of user reviews to mine valuable information for improving Apps is critical for retaining the existing users and attracting new users. Studies show that over one third of users changed their ratings following a developer response, and the median rating change is a one-star increase out of five [27]. However, analyzing user reviews face challenges. A recent study found that mobile Apps received approximately 23 user reviews per day and popular Apps, such as Facebook, received on average 4275 user reviews per day [2]. Due to the large number of user reviews, it is time-consuming, tedious, and infeasible for manual inspection to get useful opinions from the user reviews, and the unstructured and informal nature of user reviews complicates the identification of valuable information.

To this end, methods and tools are proposed and developed to automatically analyze user reviews with the aim of reducing the human effort and acquiring valuable information from the user reviews [3][9][20][24]. Chen *et al.* used a classification technique and LDA (Latent Dirichlet Allocation) [7] to mine the informative user reviews by filtering noisy and irrelevant ones from a large and rapidly increasing pool of user reviews in App markets [6]. Maalej and Nabil employed review metadata, text classification, natural language processing, and sentiment analysis techniques to automatically classify user reviews into four types: bug reports, feature requests, user experiences, and ratings [1]. Panichella *et al.* used natural language processing and sentiment analysis techniques to automatically classify user reviews into four types:

information seeking, information giving, feature request, and problem discovery, which are relevant to software maintenance and evolution [8]. Gu and Kim not only automatically classified user reviews into five types: aspect evaluation, bug reports, feature requests, praise, and others, but also summarized users' sentiments and opinions toward corresponding aspects, answering the important question "*what parts are loved by users*" for App developers [9]. Vu *et al.* proposed a keyword-based framework for semi-automated user review analysis. Their method lists the user reviews most relevant to those keywords which describe developers' interests [10].

Non-Functional Requirements (NFRs) describe a set of quality attributes that a software system should exhibit. NFRs specify a broad range of qualities such as reliability, performance. These qualities play a critical role in user experience, and should be identified and considered in App development. Moreover, NFRs are often overlooked during requirements elicitation which leads to the situation that the implemented system fails to meet users' expectation. The goal of this work is to help App developers to identify and consider NFRs in App development through automatically classifying user reviews. In this work, we combined four classification techniques BoW (Bag-of-Words), TF-IDF (Term Frequency - Inverse Document Frequency), CHI^2 (Chi Squared), and AUR-BoW (Augmented User Reviews - Bag-of-Words) with three machine learning algorithms Naive Bayes, J48, and Bagging to automatically classify user reviews into four types of NFRs (reliability, usability, portability, and performance), Functional Requirement (FR), and Others. The definition of each type is detailed in Section 3.2. We conducted experiments on the user reviews of two popular Apps (iBooks in Apple App Store and WhatsApp in Google Play Store) to compare the classification results through all the combinations of the techniques and algorithms.

The rest of the paper is organized as follows: Section 2 introduces the classification techniques used in this work in detail. Section 3 describes the research questions, the user reviews classification process with several important phases, and the dataset for the experiments. Section 4 presents the experiment results and discusses their implications. Limitation and threats to validity are discussed in Section 5. Related work is presented in Section 6. We conclude this work with further work directions in Section 7.

2 User Reviews Classification Techniques

We introduce four classification techniques BoW, TF-IDF, CHI^2, and AUR-BoW in this section, which are used for user reviews classification. BoW employs all unique terms in the user reviews of a dataset as textual features, and uses term frequency as the weight of textual features. Different from BoW, TF-IDF combines term frequency with inverse document frequency to get the weight of textual features, which is influenced by the frequency of the term in the user reviews of the dataset. Also compared with BoW, CHI^2 decreases the number of textual features, and AUR-BoW augments user reviews by most similar words. In these four classification techniques, AUR-BoW is our proposed approach, which classifies user reviews augmented by most similar words to the user reviews based on words similarity. The processing phases of user reviews classification are introduced in Section 3.

2.1 Bag-of-Words

The Bag-of-Words (BoW) model [13] is widely used to represent textual documents in information retrieval systems and is one of the most popular representation techniques for object classification. Maalej and Nabil extracted textual features and calculated the weight of textual features by BoW for user reviews classification [1]. BoW considers a dictionary which is composed of all unique terms of user reviews in the corpus as textual features, and uses term frequency (TF), the times a term appears in a user review as the weight of textual features for training classifiers. In summary, using BoW a user review j is expressed by a vector $X_j = (x_{1,j} \dots x_{i,j} \dots x_{n,j})$, in which $x_{i,j}$ denotes the weight of feature i calculated by the frequency of term i in user review j, and n denotes the number of terms in the dictionary. After that, the manually classified user reviews which are expressed by vectors act as input of supervised machine learning algorithms, and consequently are used to train classifiers.

2.2 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) uses the same textual features (i.e., words) as BoW does, but different from BoW, TF-IDF uses not only TF but also inverse document frequency (IDF) as the weight of textual features (words). TF-IDF of each word is defined in Formula (1):

$$TF - IDF(word_{i,j}) = f_{i,j} * log \frac{total\_user\_reviews}{total\_user\_reviews\_with\_word\_i} \quad (1)$$

$f_{i,j}$ denotes the frequency of a word i in user review j. We use total user reviews that contain word i as denominator since some words might appear frequently in many user reviews, which means that these words contain less type information. On the other hand, IDF alone does not consider type information, and it cannot handle the situation that a word appears in many user reviews of the same type.

2.3 Chi Squared

Motivated by IDF, we use a feature selection algorithm Chi Squared [15], which is a common statistical test and considers type information of user reviews, to select textual features important to user reviews classification from the textual features acquired by BoW. Words that exist in more types contain less type information, and consequently are less important for user reviews classification. Furthermore, many research results show that feature selection can improve the performance of text classification [5]. The frequently used feature selection algorithms are Mutual Information, Information Gain, Chi Squared (CHI^2), etc. [15]. Among them, CHI^2 is reported by many studies as one of the most effective algorithms. CHI^2 is defined in Formula (2):

$$CHI^2(t_i, C_k) = \frac{N * (ad - bc)^2}{(a + c) * (b + d) * (a + b) * (c + d)} \quad (2)$$

The variables a, b, c, and d of Formula (2) are described in Table 1. N in Formula (2) denotes the total number of user reviews in the training set. The greater the value of $CHI^2(t_i, C_k)$ is, the more type information term t_i contains. We acquire the CHI^2 value of each word in each type. After that for each type, words are ranked by the CHI^2 value in a descending order and we choose top n percent of words as textual features. After the value of n is determined, term frequency of textual features is used as the weight of the selected textual features. The value of n is a threshold which depends on specific classification experiments on the dataset. The potential value of n ranges from 1% to 100% and we use binary search method, which excludes half of the remaining possible values each time, to efficiently choose the value of n which achieves the best classification result in F-measure. For binary search, there are three variables: *begin*, *end*, and *best classification result* which denote begin of range, end of range, and best classification result in F-measure respectively. Algorithm 1 shows the execution process of binary search which returns the value of n achieving the best classification result. The time complexity of binary search is *O(logN)*, which is acceptable in practice. For Naïve Bayes, J48, and Bagging (the three machine learning algorithms used for training classifiers, see Section 3.6), the values of n of CHI^2 are set to 9%, 11%, and 17% respectively since these values achieve the best classification results in F-measure.

Table 1: Variables of Formula (2)

	Belong to category C_k	**Do not belong to category C_k**
The number of user reviews with term t_i	a	b
The number of user reviews without term t_i	c	d

```
1  begin = 1%
2  end = 100%
3  n = 100%
4  initialize best_classification_result according to  n
5  while ( begin < end)
6      n = ( begin + end ) / 2
7      get classification_result_in_F-measure according to n
8      if classification_result_in_F-measure <  best_classification_result
9          begin = n
10     else
11         end = n
12         best_classification_result = classification_result_in_F-measure
13 n = end
14 return n
```

Algorithm 1: Binary search method for getting the value of *n*.

2.4 Augmented User Reviews (the proposed approach)

Most user reviews are very short and contain less than 100 words, and user reviews are further split into sentences, which makes the text to be classified even shorter. To handle this problem, the sentences are augmented by several most similar words to the user reviews in the training set. After that, the augmented user reviews act as input of BoW, and we name this classification technique as AUR-BoW.

We use the word2vec[1] tool, which provides a vector-based representation of words to get words similarity by multiplying the vector of words. A recent study shows that word2vec provides the state-of-the-art performance for measuring words similarity [11]. The similarity between user review r_k and term t_j is calculated based on the definition in [16], which is shown in Formula (3):

$$sim(r_k, t_j) = \sum_{i=1}^{n} \left(w_{k,i} * sim(t_{k,i}, t_j)\right) \quad (3)$$

$$N = \theta * user\_review\_length \quad (4)$$

in which r_k denotes the user review k expressed by a vector $r_k = (t_{k,1} \ldots t_{k,i} \ldots t_{k,n})$, $t_{k,i}$ denotes term i in r_k, n denotes the number of terms in r_k, $w_{k,i}$ denotes the weight of term $t_{k,i}$, and $sim(t_{k,i}, t_j)$ denotes the similarity between terms $t_{k,i}$ and t_j, which is calculated by word2vec. Specially the similarity between $t_{k,i}$ and t_j is excluded when $t_{k,i}$ is equal to t_j, which increases the possibility of selecting terms not belonging to r_k when augmenting user reviews. For each user review, we traverse all unique terms in the training set to get the similarity values between terms and the user review. The top N terms in the training set ranked according to their similarity values are added to the end of the user review, which leads to an augmented user review. The value of N depends on the length of the user review and is calculated by Formula (4). θ is a threshold increasing from 0 with an increment interval of 0.1. With the increase of θ, the classification results in F-measure have no obvious tendency, making it challenging to choose the value of θ which achieves the best classification result in F-measure. To enhance the practicability of AUR-BoW, the maximum value of θ is set to 2. For Naïve Bayes, J48, and Bagging (the three machine learning algorithms used for training classifiers, see Section 3.6), the values of θ of AUR-BoW are set to 1.9, 1.1, and 1.5 respectively, which achieve the best classification results in F-measure within the set range of (0, 2). For example, a preprocessed user review is "*crash every time*" (see Section 3.5 for details about preprocessing user reviews). According to AUR-BoW when combined with J48, three words "*freezing*", "*log*", and "*everytime*" are chosen for augmenting the user review, which makes the user review contain more related information than its original form. The augmented user review for classification is "*crash every time freezing log everytime*". Different from BoW, TF-IDF, and CHI^2, AUR-BoW exploits textual semantics of user reviews, while BoW, TF-IDF, and CHI^2

[1] https://code.google.com/p/word2vec/

Table 2: Descriptions and Examples of Each Type (NFRs [35], FR, and Others)

Type	Description	Example from User Reviews
Usability	Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.	"*I have to flip through chapters to start the book.*"
Reliability	Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.	"*The app just crashes after idling for five minutes.*"
Portability	Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.	"*before my iPad worked fine but on the iphone nothing wouldn't show.*"
Performance	The extent to which a function must be executed under stated conditions.	"*Lately, I have to wait a long time for the app or a book.*"
FR	Things that a system/product should do.	"*can you plz make a screen recorder so we can make cool videos for youtube.*"
Others	Any types which are not associated with NFR and FR, e.g., emotional expressions.	"*I loved this App.*"

only consider whether a word is contained in a user review without considering the relationship between words. AUR-BoW makes use of words similarity to augment user reviews.

3 RESEARCH DESIGN

The research design of this work is described in this section in detail. We first define two research questions (RQs) in Section 3.1. We explain why we classify user reviews into four types of NFRs (reliability, usability, portability, and performance), Functional Requirements (FRs), and Others in Section 3.2. We then introduce the dataset used for experiments in Section 3.3. After that, we describe the user reviews classification process in Section 3.4. At last, we describe the important phases (Phase 2, Phase 5, and Phase 6) of the user reviews classification process in Section 3.5, 3.6, and 3.7 respectively.

3.1 Research Questions

The goal of this work is to help App developers to identify and consider NFRs in App development by automatically classifying user reviews. More specifically, we plan to study how accurately the classification techniques and machine learning algorithms can classify the user reviews into NFRs, FRs, and Others based on an NFR standard (ISO 25010) [35]. Since classification techniques and machine learning algorithms are two main components of automatic user reviews classification as detailed in Section 3.4, two research questions are formulated:

RQ1. ***Which classification technique works best (BoW vs. TF-IDF vs. CHI^2 vs. AUR-BoW) for classifying user reviews into NFRs, FRs, and Others?***

Rationale: We use four classification techniques to represent user reviews. Different from the three classification techniques BoW, TF-TDF, and CHI^2, AUR-BoW proposed in this work exploits textual semantics to augment user reviews for classification. We therefore seek to understand whether AUR-BoW works best or not in all the four classification techniques.

RQ2. ***Which machine learning algorithm works best (Naïve Bayes vs. J48 vs. Bagging) for classifying user reviews into NFRs, FRs, and Others?***

Rationale: A machine learning algorithm may lead to different performance when employed in various applications. For instance, Naïve Bayes performs best when classifying user reviews into four types (Bug reports, Feature requests, User experiences, and Ratings) in [1], while J48 gets the best results when classifying user reviews into four types (Feature Request, Problem Discovery, Information Seeking, and Information Giving) in [8]. By answering this question, we seek to understand which machine learning algorithm is most suitable for classifying user reviews into NFRs, FRs, and Others.

3.2 Types of User Reviews

User reviews are automatically classified into six types according to ISO 25010 [35], which defines quality characteristics of software systems. In ISO 25010, quality requirements are classified into eight types: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. In these quality requirement types, functional suitability considers the meta level of requirements, while user reviews normally elaborate concrete requirements; maintainability is about the internal quality, while user reviews concern more about the external quality of a system; and security and compatibility are not found in our dataset (see Section 3.3). For these reasons, we consider four types of NFRs in ISO 25010: usability, reliability, portability, and performance, with FR and Others as the types for the classification of user reviews. The descriptions and examples of these types are provided in Table 2.

3.3 Experiment Material

We evaluated the combination of the classification techniques and machine learning algorithms on a set of user reviews. We selected two popular Apps (iBooks in the books category from Apple App Store and WhatsApp in the communication category from Google Play) as the cases for experimentation. We collected 6696 raw user reviews from iBooks and 4400 raw user reviews from WhatsApp as the dataset. 21969 user review sentences obtained from the dataset are used for training word2vec. For each App, 2000 user review sentences were randomly sampled and manually classified, which are 4000 user review sentences in total and act as the ground truth for the evaluation (the processing process from raw user reviews to user review sentences is detailed in Section 3.4). All sampled user review sentences were manually classified by three researchers, the two authors and a master student in software engineering. We first conducted a pilot classification of 100 user review sentences from each App (200 user review sentences in total) by the three researchers independently through following an NFR standard (ISO 25010 [35]), and any disagreements on the classification results were discussed and resolved by the three researchers. After reaching a consensus on the classification of user review types, the first author and the master student classified the remaining user review sentences independently and the agreement between them is 88% (3523/4000). After that any disagreements on labeled sentences were discussed and confirmed with the second author. At last, the agreement between the first author and the master student achieved 100%. Table 3 shows the numbers and percentages of manually labeled user review sentences in the dataset that were classified as a certain type.

Table 3: Numbers and Percentages of Manually Labeled User Review Sentences in the Dataset

Type	#Sentence	Proportion
Usability	432	0.108
Reliability	587	0.147
Portability	119	0.029
Performance	121	0.030
FR	558	0.140
Others	2183	0.546
Total	4000	1.000

3.4 User Reviews Classification Process

Fig. 1 shows the execution process we followed and the techniques and algorithms we employed to automatically classify user reviews. Specially, the process is composed of six phases (***Phase 6*** is about the evaluation of the classifiers):

Phase 1: Input User Reviews: Collect user reviews from App Store by an open source tool[2] and Google Play Store.

[2] https://github.com/oklahomaok/AppStoreReview

Phase 2: Preprocess User Reviews: First, we spilt each user review collected in ***Phase 1*** into sentences as the units to be classified. Then we manually analyze each sentence and classify their types (NFR types, FR, or Others). After that we preprocess user review sentences by eliminating stop words, and carrying out lemmatization and stemming.

Phase 3: Select Textual Features and Calculate the Weight of Textual Features: We apply three classification techniques BoW, TF-IDF, and CHI^2 to extract textual features and calculate the weight of features, and we use these textual features and the weight of features to train classifiers.

Phase 4: Augment User Reviews: We exploit the textual semantics of user reviews through calculating the similarity between words and user reviews to augment the user reviews.

Phase 5: Train Classifiers: We use three machine learning algorithms, Naïve Bayes, J48, and Bagging separately to train classifiers, and after that user reviews are automatically classified by the trained classifiers.

Phase 6: Evaluate Trained Classifiers: We use precision, recall, and F-measure to evaluate the performance of the classifiers which are trained in ***Phase 5***.

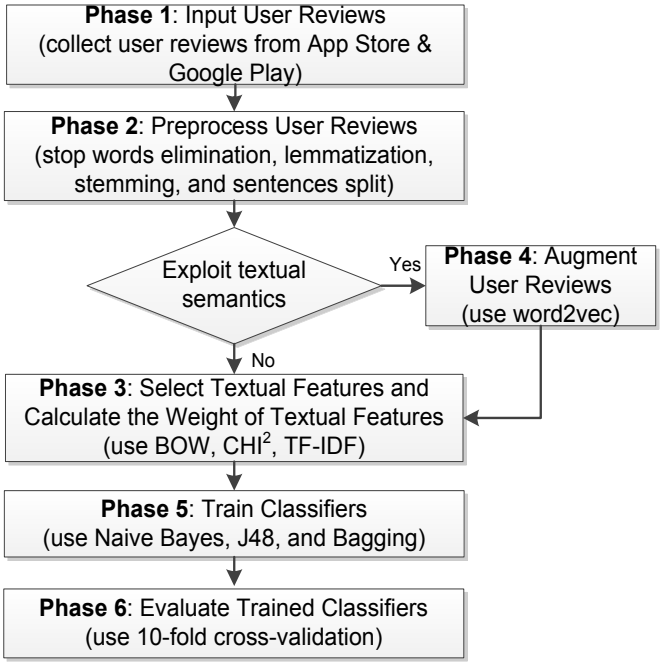

Figure 1: Processing phases of user reviews classification.

3.5 Preprocess User Reviews

User reviews preprocessing is composed of four steps: removing stop words, lemmatization and stemming, splitting user reviews into sentences, and transforming slang words or abbreviations into their basic forms. (1) Stop words usually refer to the most common words such as "*is*", "*a*", and "*at*", which do not influence the semantics of a user review. As there is no single universal list of stop words used by all natural language processing tools, we remove the words with a length of less than three letters. (2) In

English, words appear in several inflected forms for grammatical reasons but have the same meaning, such as the verb "*walk*" may appear as "*walks*", "*walked*", and "*walking*". The goal of both stemming and lemmatization is to reduce inflectional forms, but they differ in their flavor. Lemmatization takes the morphological analysis and linguistic context of the term into consideration, while stemming usually refers to a crude heuristic process that cuts out the end of words. For example, if confronting with the token "*better*", stemming might just return "*better*", whereas lemmatization would attempt to return either "*good*" or "*better*" depending on the context. If confronting with the token "*crashes*", lemmatization might return "*crashes*" according to the context, whereas stemming would return "*crash*". In summary, we combine both lemmatization and stemming to get the basic form of a word. (3) A recent study found that up to 30% of user reviews raise various types of issues in a user review [12]. NFRs and FRs are specific issues raised in user reviews, and to classify the content of user reviews in better granularity into NFRs, FRs, and Others, we split user reviews into sentences as the unit of classification. (4) Occasionally users express their opinions with slang words or abbreviations, such as "*idk*" means "*I would like*" and "*fav*" means "*favorite*". To address this issue, after splitting user reviews into sentences, we transform these words into their basic forms using a continuously updated table of slang words and abbreviations.

3.6 Train Classifiers

We used four classification techniques BoW, TF-IDF, CHI^2, and AUR-BoW introduced in Section 2 to extract textual features, which act as the input of machine learning algorithms for training classifiers. Specially, we conducted experiments using Weka [3] with the three machine learning algorithms, Naive Bayes, J48, and Bagging. The reason that we chose these algorithms for user reviews classification is that they had been successfully employed for object classification in many previous works, e.g., [1][17][32].

3.7 Evaluation Methodology

We used precision, recall, and F-measure (see Formula (5), (6), and (7)) which are commonly used in performance evaluation of information retrieval, and weighted average (see Formula (8)) which is used in calculating user reviews classification results (e.g., [8]) to measure the performance of classification results. $Number_i$ in Formula (8) denotes the number of user review sentences in $type_i$ in the testing set.

In Formula (5) and (6), TP_i denotes the number of instances classified as type i and actually are of type i; FP_i denotes the number of instances classified as type i but actually are of type j where $i \neq j$; FN_i denotes the number of instances classified as type j and actually are of type i where $i \neq j$. We performed a 10-fold cross-validation on the dataset, and each fold contained 400 user review sentences.

[3] http://www.cs.waikato.ac.nz/ml/weka/

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \quad (5)$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \quad (6)$$

$$F - measure_i = \frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i} \quad (7)$$

$$Weighted\ average(Precision/Recall/F - measure) = \frac{\sum_{i \in type}(precision_i/recall_i/F - measure_i) * Number_i}{total\_review\_sentences} \quad (8)$$

4 RESULTS ANALYSIS

In this section we present the experiment results and discuss their implications. Table 4 provides an overview of the experiment results obtained through the four classification techniques BoW, TF-IDF, CHI^2, and AUR-BoW combined with three machine learning algorithms Naïve Bayes, J48, and Bagging, which show the weighted average precision, recall, and F-measure of the classification results. In general, the precision, recall, and F-measure of all the combinations are higher than 0.644. To concentrate on a single variable for answering RQ1 and RQ2, we analyzed the classification results achieved by combining Naïve Bayes with the four classification techniques (we did the same for J48 and Bagging) and combining AUR-BoW with the three machine learning algorithms. We present and analyze the results of RQ1 and RQ2, and discuss the classification results of different types in this section.

(1) RQ1. Classification techniques

For answering RQ1, we found that the highest F-measure (0.718) is achieved by AUR-BoW which makes use of word2vec to exploit textual semantics to augment user reviews. From Table 4, it can be found that there is no much difference between TF-IDF and CHI^2. When using Naïve Bayes, compared with BoW, the F-measures achieved with TF-IDF, CHI^2, and AUR-BoW are increased by 0.61%, 1.07%, and 4.74% respectively. When using J48, compared with BoW, the F-measures achieved with TF-IDF, CHI^2, and AUR-BoW are increased by 0.75%, 0.75%, and 2.86% respectively. When using Bagging, compared with BoW, the F-measures achieved with TF-IDF, CHI^2, and AUR-BoW are increased by 0.57%, 0.29%, and 4.06% respectively.

In the four classification techniques, AUR-BoW works best for classifying user reviews into NFRs, FRs, and Others. This finding can be attributed to the word2vec tool, which exploits surrounding words in a user review sentence, and maps words with similar meanings to similar vectors. When using all the three machine learning algorithms, compared with BoW, the F-measure achieved with AUR-BoW is increased much higher than that achieved with TF-IDF and CHI^2. This finding indicates that for user review sentences, it is effective to improve user reviews classification results by adding textual semantics to the sentences. One explanation is that for short text like user review sentences, extending them by several most similar words can augment the

Table 4: Results of the Combinations of One Classification Technique from BoW, TF-IDF, CHI^2, and AUR-BoW with One Machine Learning Algorithm from Naïve Bayes, J48, and Bagging

	BoW			TF-IDF			CHI^2			AUR-BoW		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Naïve Bayes	0.665	0.644	0.654	0.668	0.649	0.658	0.669	0.654	0.661	**0.720**	0.653	0.685
J48	0.656	0.674	0.665	0.661	0.678	0.670	0.661	0.680	0.670	0.677	0.690	0.684
Bagging	0.685	0.694	0.690	0.689	0.699	0.694	0.688	0.697	0.692	0.714	**0.723**	**0.718**

information of short text, and consequently provide more information to improve classification results.

(2) RQ2. Machine learning algorithms

For AUR-BoW, the results in Table 4 show that the highest precision and recall are achieved by Naïve Bayes (0.720) and Bagging (0.723) respectively. Overall, for the balance between precision and recall, i.e., F-measure, the Bagging algorithm works best (0.718), which shows that Bagging is a more suitable machine learning algorithm for NFRs classification from user reviews than Naïve Bayes and J48.

The best F-measure (0.718) got in our experiment is similar to the best F-measure (0.720) of the experimental results produced by Panichella *et al.* in [8], which classified user reviews into four types (Feature Request, Problem Discovery, Information Seeking, and Information Giving). The machine learning algorithm J48 gets the best F-measure in [8], while in our experiments Bagging performs best (and J48 gets the worst F-measure), which indicates that for various classification applications (i.e., classified into different types), the best performance can be produced by different machine learning algorithms. The possible reason is that user reviews are classified into different types, which have different textual features. For example, feature requests mainly focus on missing functionality of an App, while usability NFRs mainly focus on whether the App can satisfy users.

(3) Classification results of different types

We further analyze the classification results of different types achieved by the combination of AUR-BoW with Bagging, which gets the best F-measure in the experiments. Table 5 shows the precision, recall, and F-measure for each type obtained by the combination of AUR-BoW with Bagging. We notice that there is an obvious difference between the F-measures of the classification results of different types ranging from 0.335 to 0.822. This finding can be attributed to that there are very few portability (119/4000, 2.9%) and performance (121/4000, 3.0%) NFRs in the user review sentences (i.e., the dataset of the experiments) compared with other types, which makes the textual features of these two NFR types for training classifiers are less than that of other types.

The F-measure of usability is higher than that of reliability and FR. This finding can be attributed to the reason that reliability and FR have lots of common words which makes it difficult to distinguish between reliability and FR. For example, the word "*download*", in a reliability NFR: "*I bought a book on august 12 and it hasn't downloaded yet.*" which means that the normal function of downloading books was broken down; in a FR: "*I downloaded some books individually on each device and the bookmarks won't sync either.*" which denotes that iBooks App provided no synchronization function of the bookmarks. Fig. 2 shows the comparison between the F-measure and proportion of each type in the dataset. The comparison indicates that the trend of the classification results of different types largely follows the trend of the proportions of the types in an unbalanced dataset (i.e., the dataset of the experiments). Automatic classification performs worse when the size of certain types is smaller (e.g., portability and performance), while there is no clear trend when the size of the types are similar (e.g., usability and reliability). This finding is interesting because it suggests that to some extent, increasing the size of certain types in the experiment dataset can improve the classification results of the types. This hypothesis should be further investigated with new experiments.

Table 5: Percentages of Classified Sentences in the Dataset and Classification Results of Various Types Obtained by the Combination of AUR-BoW and Bagging

Type	Proportion	Precision	Recall	F1
Usability	0.108	0.757	0.565	0.647
Reliability	0.147	0.595	0.552	0.572
Portability	0.029	0.632	0.327	0.431
Performance	0.030	0.596	0.233	0.335
FR	0.140	0.630	0.587	0.608
Others	0.546	0.770	0.881	0.822
Weighted Average	1.000	0.714	0.723	0.718

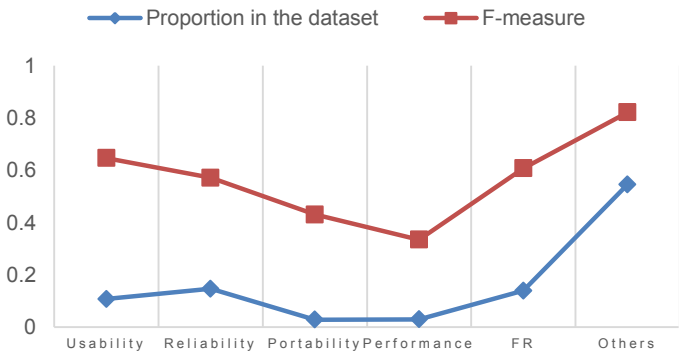

Figure 2: Comparison between the F-measure and proportion of each type in the dataset.

5 LIMITATION AND THREATS TO VALIDITY

In this section, we discuss the limitation and threats to validity in this work according to the guidelines in [33], and how these threats were partially mitigated in this work.

Limitation refers to the issues which currently cannot be addressed by our approach. We labeled each user review sentence with one type, but in reality one user review sentence may belongs to more than one type of NFRs or FR. During the process of labeling user review sentences, we found that the proportion of the sentences with multiple types is 1.1% (44/4000) in our dataset. We manually separated these sentences into more fine-grained sentences which can be labeled in one type before conducting the classification experiments. For instance, the sentence "*it uses more battery power, i know i can have a custom black background on my conversations but not on contacts and groups list, can't you guys create an option for that on a future update*" should be classified as performance NFR and FR, we separated this user review into two sentences, one is "*it uses more battery power*" which was classified as performance NFR and the other is "*i know i can have a custom black background on my conversations but not on contacts and groups list, can't you guys create an option for that on a future update*" which was classified as FR. But we admit that this is a limitation of our approach which cannot attach multiple labels to one user review sentence.

Construct validity focuses on whether the theoretical constructs are interpreted and measured correctly. A threat to construct validity in this study involves whether the user review sentences used for the experiments were classified correctly by the researchers. To achieve a common understanding of various NFR types, we used the definitions of NFR types in the ISO 25010 standard [35]. But using a standard cannot guarantee that the researchers understand the definitions of various NFR types in the same way, and another risk is that whether the researchers classify the user review sentences from iBooks and WhatsApp into the types which they actually belong to. A pilot classification of 100 user review sentences from each App (200 sentences in total) was conducted by three researchers independently, and any disagreements on the classification results were further discussed and resolved by the three researchers, in order to get a consensus among researchers on the classification of user review types. The remaining of the user review sentences were manually classified by the first author and a master student in software engineering independently, and any disagreements on labeled sentences were discussed with the second author. These measures were used to partially mitigate personal bias in user reviews classification. Another threat to the construct validity in this study is whether the user reviews used in the experiment are sufficient to draw reasonable conclusions. To mitigate this threat, we used a random sample of collected user reviews.

Internal validity focuses on the design of a study, especially whether the results follow from the data. One threat to the internal validity in this study is the tests overfitting of the machine learning [37]. To mitigate the influence of this threat, we applied a 10-fold cross-validation in our experiments.

External validity refers to the degree to which our findings from this study can be generalized in other settings. To mitigate this threat, we conducted the experiments on the user reviews of two popular Apps: iBooks (iOS) in the books category and WhatsApp (Android) in the communication category. The diversity of the chosen Apps and their platforms increases the generalizability of the experiment results and decreases the potential sampling bias. But it is still unclear whether this experiment can attain similar results when being applied to other categories of Apps (e.g., Facebook). Another threat to external validity is that two types of NFRs, security and compatibility, were not available in the experiment dataset, and there are very few portability and performance NFRs in the dataset. We plan to conduct a large-scale empirical study through collecting more user reviews from diverse Apps, in order to improve the external validity in the next step.

Reliability refers to whether the study yields the same results if other researchers replicate this study, which in this work is related to the processes of automatic user reviews classification (the process of the experiments) and manual user reviews classification (the process for producing the experiment dataset). By making explicit these processes (as detailed in Section 3.3 to Section 3.6), we believe that this threat is partially mitigated.

6 RELATED WORK

We summarize and discuss the related work on user reviews analysis in this section, including user reviews filtering, classification, and summarization.

In recent years, App Store analysis has become a popular topic in software engineering [28], and users can easily submit their feedback about the Apps they used in App platforms. Al-Subaihin *et al.* extracted features from textual description of Apps in order to cluster the Apps using agglomerative hierarchical clustering and produced an effective categorization of Apps [30]. Tian *et al.* analyzed the characteristics of user reviews and identified the characteristics of high-rated Apps [29]. Study shows that user analytics tools will help developers to deal with the large numbers of user feedback (e.g., user reviews) by filtering, classifying, and summarizing them, to decide what requirements and features they should add, change, or eliminate [18].

App reviews filtering has drawn increasing attention in the software engineering community since only 1/3 of user reviews are useful for developers [8]. Oh *et al.* proposed an algorithm that automatically identifies informative reviews reflecting user involvement to reduce the information overload of developers [21]. Chandy and Gu proposed an approach to automatically identify spam user reviews in App Store using a latent class model with interpretable structure and low complexity [4]. Our work is different from these works in that we try to employ supervised machine learning algorithms to not only filter non-informative user reviews but also classify user reviews into six types (four types of NFRs, FRs, and Others).

Recently, a number of approaches have been proposed for automatically classifying user reviews. Yang and Liang combined TF-IDF and regular expression (an NLP technique) with human

intervention to classify user reviews into FRs and NFRs [38]. Panichella *et al.* merged three techniques: natural language processing, text analysis, and sentiment analysis, to automatically classify user reviews into four types: feature request, problem discovery, information seeking, and information giving [8]. Villarroel *et al.* not only classified user reviews into suggestion for new features, bug reports, and other types, but also clustered together related user reviews and recommended the user review cluster developers should satisfy in the next release [39]. Hoon *et al.* developed three ontologies: emotion ontology, functional ontology, and quality ontology to classify user reviews into three categories which follow the three ontologies [36]. Maalej *et al.* and Blei *et al.* analyzed user reviews at the sentence level [1][8], while McIlroy *et al.* presented an approach at the user review level that can automatically assign multi-labels to user reviews [12]. Similar with Chen and Kao's work which combined the word co-occurrence information with Bi-term topic model (BTM) to augment Topic Model and made Topic Model more appropriate for shorter text like tweets [40], in this work we not only split user reviews into sentences, but also augmented user review sentences by calculating the similarity between terms and user review sentences.

A number of researchers have focused on mining and analyzing textual data with the goal of deriving important information (e.g., NFRs) to help developers maintain and evolve their software systems. Sorbo *et al.* summarized thousands of user reviews to recommend future software changes according to the summary of user reviews [22]. Rastkar *et al.* produced bug report summaries which help developers to save time in detecting duplicate bug reports [23]. Gao *et al.* developed a tool AR-Tracker, which automatically collects user reviews of Apps and ranks them in order to optimize the representation of the reviews set [19]. Guzman *et al.* used a feature and sentiment centric approach to extract different opinions and experiences about using Apps from user reviews [41]. Other researchers proposed various approaches to automatically classify non-functional requirements from requirements specifications through information retrieval [25], clustering [26], and text mining techniques [34] respectively. In this work, we focus on classifying NFRs from user reviews, which are short and unstructured compared with requirements specifications.

7 CONCLUSIONS AND FUTURE WORK

In this work, we combined four classification techniques with three machine learning algorithms to automatically classify user reviews into four types of NFRs (reliability, usability, portability, and performance), FRs, and Others. Specially, we exploited textual semantics to augment user reviews by word2vec for automatically classifying user reviews. We evaluated the combinations of the classification techniques and machine learning algorithms with user reviews collected from two popular Apps: iBooks and WhatsApp which belong to different categories (domains) from different App stores (platforms). We conducted experiments to compare the F-measure of the classification results through all the combinations. We found that the combination of AUR-BoW with Bagging achieves the highest F-measure of 71.8%, with a precision of 71.4% and a recall of 72.3% respectively. This finding shows that augmented user reviews can improve the results of user reviews classification. Moreover, in an unbalanced dataset, automatic classification performs worse when the size of certain types is obviously smaller. The automatic classification of NFRs from user reviews can help App developers to better understand user reviews and meet user needs from an NFR perspective, which is meaningful for developers to retain the existing users and attract new users.

In the next step, the approach AUR-BoW, which achieves the best classification results on NFRs, can be improved in two promising aspects:

(1) To validate AUR-BoW with the user reviews of other categories of Apps (e.g., Facebook) in order to improve the external validity of the results.

(2) To combine AUR-BoW with other classification techniques (CHI^2, TF-IDF), which may achieve better results for user reviews classification.

(3) Two types of NFRs, security and compatibility, do not appear in the experiment dataset, and there are very few portability and performance NFRs in the dataset. We plan to collect user reviews from more Apps, which contain various types of NFRs for replicating the experiment. This is another aspect to mitigate the threats to the external validity: whether the results hold for other types of NFRs.

(4) The types is imbalanced in the experiment dataset. There are very few portability (119/4000, 2.9%) and performance (121/4000, 3.0%) NFRs in the user review sentences. We plan to use techniques (e.g., [42]) to mitigate this imbalance in the dataset, which might improve the accuracy of types with fewer training data.

ACKNOWLEDGMENTS

This work has been partially sponsored by the Natural Science Foundation of China (NSFC) under Grant No. 61472286. We would like to thank Tianlu Wang, who participated in the manual classification of the user review sentences for the experiments.

REFERENCES

[1] W. Maalej and H. Nabil. 2015. Bug report feature request or simply praise? On automatically classifying app reviews. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference* (RE'15). IEEE, 116-125.

[2] D. Pagano and W. Maalej. 2013. User feedback in the appstore: an empirical study. In *Proceedings of the 21st IEEE International Requirements Engineering Conference* (RE'13). IEEE, 125-134.

[3] C. Iacob and R. Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceeding of the 10th IEEE Working Conference on Mining Software Repositories* (MSR'13). IEEE, 41-44.

[4] R. Chandy and H. Gu. 2012. Identifying spam in the IOS app store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality* (WebQuality'12). ACM, 56-59.

[5] Y. Yang and J. P. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning* (ICML'97). Morgan Kaufmann, 412-420.

[6] N. Chen, J. Lin, Steven C. H. Hoi, X. Xiao, and B. Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering* (ICSE'14). ACM, 767-778.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3, (2003), 993-1022.

[8] S. Di Panichella, A. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution* (ICSME'15). IEEE, 281–290.

[9] X. Gu and S. Kim. What parts of your apps are loved by users? 2015. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering* (ASE'15). IEEE, 760-770.

[10] P. M. Vu, T. T. Nguyen, and H. V. Pham. 2015. Mining user opinions in mobile app reviews: a keyword-based approach. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering* (ASE'15). IEEE, 749-759.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop of 1st International Conference on Learning Representations* (ICLR'13).

[12] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empirical Software Engineering 21, 3 (2016), 1067-1106.

[13] Y. Zhang, R. Jin, and Z. H. Zhou. 2010. Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics 1, 1-4 (2010), 43-52.

[14] P. Liang, P. Avgeriou, K. He, and L. Xu. 2010 From collective knowledge to intelligence: pre-requirements analysis of large and complex systems. In *Proceedings of the 1st Workshop on Web 2.0 for Software Engineering* (Web2SE'10), ACM, 26-30.

[15] G. Forman. 2003. An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research 3, 3 (2003), 1289–1305.

[16] C. H. Li, J. C. Yang, and S. C. Park. 2012. Text categorization algorithms using semantic approaches corpus-based thesaurus and WordNet. Expert Systems with Applications 39, 1 (2012), 765-772.

[17] Y. Zhou, Y. Tong, R. Gu and H. Gall. 2014. Combining text mining and data mining for bug report classification? In *Proceedings of the 30th IEEE International Conference on Software Maintenance and Evolution* (ICSME'14). IEEE, 311-320.

[18] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. 2016. Toward data-driven requirements engineering. IEEE Software 33, 1 (2016), 48–54.

[19] C. Gao, H. Xu, J. Hu, and Y. Zhou. 2015. Ar-tracker: track the dynamics of mobile apps via user review mining. In *Proceedings of the 10th IEEE Symposium on Service-Oriented System Engineering* (SOSE'15). IEEE, 284–290.

[20] S. Xie, G. Wang, S. Lin, and P. S. Yu. 2012. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD'12). ACM, 823-831.

[21] J. Oh, D. Kim, U. Lee, J. G. Lee, and J. Song. 2013. Facilitating developer-user interactions with mobile app review digests. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems* (CHI'13). ACM, 1809-1814.

[22] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. Gall. 2016. What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (FSE'16). ACM, 499-510.

[23] S. Rastkar, G. C. Murphy, and G. Murray. 2014. Automatic summarization of bug reports. IEEE Transactions on Software Engineering 40, 4 (2014), 366-380.

[24] L. V. Galvis Carreno and K. Winbladh. 2013. Analysis of user comments: an approach for software requirements evolution. In *Proceedings of the 35th International Conference on Software Engineering* (ICSE'13). IEEE, 582-591.

[25] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. 2007. Automated classification of non-functional requirements. Requirements Engineering 12, 2 (2007), 103-120.

[26] A. Mahmoud and W. Grant. 2016. Detecting classifying and tracing non-functional software requirements. Requirements Engineering 21, 3 (2016), 1-25.

[27] S. McIlroy, W. Shang, N. Ali, and A. Hassan. 2015. Is it worth responding to reviews? A case study of the top free apps in the Google Play store. IEEE Software, DOI: 10.1109/MS.2015.149

[28] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. 2016. A Survey of app store analysis for software engineering. IEEE Transactions on Software Engineering, DOI= 10.1109/TSE.2016.2630689

[29] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan. 2015. What are the characteristics of high-rated apps? A case study on free Android applications. In *Proceedings of the 31th IEEE International Conference on Software Maintenance and Evolution* (ICSME'15). IEEE, 301-310.

[30] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. ZhangTavecchia. 2016. Clustering mobile apps based on mined textual features. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM'16). ACM, 1-38.

[31] Number of apps available in leading app stores as of June 2016, http://www.statista.com/statistics/276623/number-of-apps-available-inleading-app-stores/, accessed on 2016-07-01.

[32] J. R. Quinlan. 1996. Bagging boosting and C4.5. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence* (AAAI'96). AAAI Press, 725-730.

[33] F. Shull, J. Singer, and D. I. Sjøberg. 2008. Guide to advanced empirical software engineering. Springer-Verlag, London, DOI: 10.1007/978-1-84800-044-5

[34] W. Zhang, Y. Yang, Q. Wang, and F. Shu. 2015. An empirical study on classification of non-functional requirements. In *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering* (SEKE'15). Knowledge Systems Institute, 190-195.

[35] ISO, ISO/IEC 25010, 2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. In ISO/IEC FDIS 25010, 2011, 1-34.

[36] L. Hoon, M. A. Rodriguez-García, R. Vasa, R. Valencia-García, and J. G. Schneider. 2016 App reviews: breaking the user and developer language barrier. *In Trends and Applications in Software Engineering*. Springer International Publishing, 223-233.

[37] T. Dietterich. 1995. Overfitting and undercomputing in machine learning. ACM computing surveys 27, 3 (1995), 326-327.

[38] P. Liang and H. Yang. 2015. Identification and classification of requirements from app user reviews. In *Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering* (SEKE'15). Knowledge Systems Institute, 7-12.

[39] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering* (ICSE'16). ACM, 14-24.

[40] G. B. Chen and H. Y. Kao. 2015. Word co-occurrence augmented topic model in short text. International Journal of Computational Linguistics and Chinese Language Processing 20, 2 (2015), 45-64.

[41] Emitza Guzman, Omar Aly, and Bernd Bruegge. 2015. Retrieving diverse opinions from app reviews. In *Proceedings of the 9th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM'15). ACM, 1–10.

[42] B. Wallace, K. Small, C. Brodley, and T. Trikalinos. 2011 Class imbalance, redux. In *Proceedings of the 11th IEEE International Conference on Data Mining* (ICDM'11). IEEE, 754–763.