SatisIndicator: Leveraging User Reviews to Evaluate User Satisfaction of SourceForge Projects

Zhenzheng Qian*, Beijun Shen†, Wenkai Mo‡ and Yuting Chen§
*†‡§School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai, China
Email: bjshen@sjtu.edu.cn

***Abstract*—The quality of a software project is important for users, as it may incur high cost when a user or a company happens to pick up a software project with low quality. In recent years, many software quality assessment models start to take user satisfaction as an important metric for measuring the software quality. However, user satisfaction on a software project may not be evaluated precisely. In this paper, we proposed a novel approach called *SatisIndicator* to automatically evaluate the user satisfaction from user reviews which implicate user opinions. The essential idea of SatisIndicator is to (1) use a topic model to cluster reviews of a software genre into different topics and calculate their weights, (2) take sentiment analysis to calculate the sentiment strength of user attitudes, and (3) match attitudes with corresponding topics and calculate user satisfaction. SatisIndicator also applies Wilson Interval to punish the software with few reviews in order to keep fairness during evaluation. We have evaluated SatisIndicator on ten software genre datasets on SourceForge.net. The evaluation results show that when softwares have both sufficient and insufficient reviews, SatisIndicator performs 35% higher than baselines at p@3, 15% higher than baselines at p@15 and over 85% Spearman Coefficiend. When softwares have insufficient reviews, SatisIndicator performs 30% higher than baselines at p@3, 15% higher than baselines at p@15 and over 60% Spearman Coefficiend.**

***Keywords*—User satisfaction, Sentiment analysis, User review analysis, Topic model**

I. Introduction

Many software projects are accessed by end-users, programmers and commercial companies, for completing specific computation tasks and/or speeding up their software development. Meanwhile, the quality of the software projects is still a threat to the use of these projects: it may incur high cost when a user or a company happens to pick up a software project with low quality. Several works have been done to evaluate source codes like: Code measurement [1], [2], [3]. It extracts code features that high quality software project products source codes have, then evaluates the quality of a new software product using these features.

In this paper, we concentrate on evaluating the user satisfaction from user reviews. As in some well-known software quality models (e.g., the quality-in-use model in ISO/IEC25010 [4]), user satisfaction is the most difficult characteristic to evaluate automatically.

Some challenges do exist when we evaluate user satisfaction:

Challenge1: A user review may have several attitude aspects, and different attitude aspects have different importance degree

We cannot view them as the same. It is a hard job to find out which topic an attitude aspect belongs to and to weight topics. We use topic model to cluster all reviews of a software genre, to get topics that users care about and the weight of each topic.

Challenge2: User reviews are subjective data, it is hard to evaluate them quantitatively.

Traditional sentiment analysis [5], [6], [7], [8], [9] can be adopted to partly overcome this challenge. But to completely overcome the challenge, we need to formulate the problem of calculate user satisfaction.

Challenge3: About 67% softwares only have one or two reviews.

It is hard to evaluate user satisfaction with few reviews. Because if there are few reviews can be used to analyze, the confidence level will decrease. For instance, a software with one hundred reviews and 50% of them are positive will has higher confidence level than a software with two reviews and 50% of them are positive. Wilson Interval is used to punish the software with insufficient reviews.

In this paper we proposed a novel approach, *SatisIndicator*, to evaluating user satisfaction from user reviews automatically. User reviews implicate the attitude information of the users. Furthermore, almost all the software project download platforms provide comment function to users. Users can write down their subjective attitudes of the software project on the platform after they downloaded and experienced it. What we do is to design a fine grain approach to evaluate the user satisfaction and make the results close to what human evaluate. First, topic model is applied to calculated the weight of user attitudes. Then, improved recursive neural tensor network is used to help analyzing sentiment strength of user attitudes. Finally, we compute the user satisfaction using attitudes sentiment strength times corresponding weight. And Wilson Interval is applied to punish the software with insufficient reviews. Since SatisIndicator use user attitude aspect rather than user review as the smallest unit to calculate the user satisfaction, the result will be more similar to which is calculated with by human.

This paper makes the following contributions:

1) We formulate the problem of calculating user satisfaction from reviews. Which makes user satisfaction can be calculated automatically like objective data.
2) We present SatisIndicator as a novel approach to tackle this problem, which evaluate user satisfaction from a fine grain of every user attitude aspect.

3) We do exhaustive experiments on ten software genres in SourceForge, in which empirical results show that SatisIndicator has 35% higher than baselines at p@3, 15% higher than baselines at p@15 and over 85% Spearman Coefficient. And when softwares have few reviews SatisIndicator has 30% higher than baselines at p@3, 15% higher than baselines at p@15 and over 60% Spearman Coefficient.

The remainder of this paper will be as follow. We will talk about the overview of our approach and the detail realization phases in the second section. The third section is experiment, we are going to show the accuracy of SatisIndicator, and the performance on real data set. In the forth section, we will talk about some related works. The conclusion is at section five.

II. Approach

A. Overview

In this section, we will show the overview of our approach. We divide our approach into three phases in Fig.1: 1) In the first phase, we use natural language processing method to process user reviews, classify the same genre's reviews into informative reviews and non-informative reviews, use topic model to cluster informative reviews in the same software genre to get topics that users care about, and calculate the weight of each topic. 2) In the second phase, for a particular software reviews, we extract attitude aspects and discover pure attitude reviews and calculate their sentiment strength. 3) In the third phase, we match attitude aspects and pure attitude reviews with topics, calculate the user satisfaction of a software using what attitude's sentiment strength times corresponding topic weight, and apply Wilson Interval to punish softwares with few reviews.

B. Computing the Weight of Every Topic

Pre-processing data. Before using the reviews data, we need do some regular nature language processing (NLP) on them. The pre-process including: 1)*tokenization:* split sentence or document into words. 2)*To lower case:* transfer all words to lower case. 3)*Stopword removal:* delete stopword like "it", "is", "and", "I". 4)*Lemmatization:* reduce different inflected forms of a word to their basic lemma. For example, transfer "working", "worked", "works" to "work". 5)*Finding high-frequency bigrams:* add an underline between two adjacent words which co-occur frequently but have different meanings when they show up alone. An example is that words "open" and "source" are co-occurrence in high frequency, but they respectively has different meaning when used alone. We can add an underline between them and make them a new word "open_source". We ignore bigrams appearing less than 3 times.

Identifying informative Reviews. In software project community, we can get lots of user reviews. But not all of them are useful. We define advertisements, and other irrelevant texts containing no attitude (e.g., questions, answers, and random texts) as non-informative reviews. For example, advertisements like*"what Jeffrey replied I am amazed that a mother can profit 9732 Dollars in one month on the computer. did you look at this link (contact numbers also available on home page) Lazy-Cash9.Com"*, questions like *"How can I play mp4 format"*, answers like *"The blue button which is besides the logo."* and random texts like *"kshfabfjshgdjvbg"* are non-sense to our work, sometimes even have negative effect on our topic results. So we need to filter non-informative reviews out of informative reviews. We use the Naive Bayes classifier based on bigrams realized by *lingpipe*[1] (since it has the highest precision among other classifiers in lingpipe) to do this. There is another kind of special reviews which also belong to informative reviews but has no attitude aspects, called pure attitude reviews For example, "Good job!"). Pure attitude reviews are about 40% in all software reviews in SourceForge.net.

Clustering User Reviews. We need to find out which topics users care about and weight them. We also need topics that have semantic relation rather than just literal repetition. For example *"The video player can only play a little format videos"* and *"It can open the mp4 format."*, traditional clustering methods using cosine distance can only capture the explicit similarity between them, like "format". In order to further explore the implicit semantic relations between sentences, we apply topic model to get the topic representation of each sentence.

There are two main types of topic model, Probabilistic Latent Semantic Indexing (pLSI) [10] and Latent Dirichlet Allocation (LDA) [11]. We use LDA to cluster user reviews, because LDA can solve problems of over fitting and difficult assigning probability to a document outside of the training set in pLSI.

We can view LDA as a three layers Bayes model. A document is generated in the LDA model as follow:

1) Choose θ_j, a topic-document Multinomial distribution from the Dirichlet distribution on hyper parameter α. $j \in \{1, ..., M\}$, M is the number of reviews.
2) Choose ϕ_k, a word-topic Multinomial distribution from the Dirichlet distribution on hyper parameter β. k$\in \{1, ..., K\}$, K is the number of topics.
3) For each of the word position j, i where $j \in \{1, ..., M\}$, and $i \in \{1, ..., N_j\}$, N_j is the number of word in review j. Iteratively choose a topic $z_{j,i}$ from Multinomial distribution(θ_j) and choose a word $w_{j,i}$ from Multinomial distribution($\phi_{z_{j,i}}$).

Since α, β can be set by us and every word in a review is known, we can get two distributions (review-topic distribution $P(Z|R)$ and topic-word distribution $P(W|Z)$) by using LDA. The number of topics will impact the topic weight with a great degree, we must make sure it is suitable. We use the method mentioned in [12] to choose the suitable number of topics. Calculate the log-likelihood by changing the number of topics T and choose the k as topic number when log-likelihood reaches a peak at $T = k$ (as Eq.(1) shown).

$$\log P(W|T) \approx \log \prod_{i=1}^{N} P(w_i) = \sum_{i=1}^{N} \log \sum_{k=1}^{T} P(w_i|z_k)P(z_k) \tag{1}$$

$$P(z_k) = \frac{\sum_{j=1}^{M} P(z_k|r_j)}{M} \tag{2}$$

[1] http://alias-i.com/lingpipe/

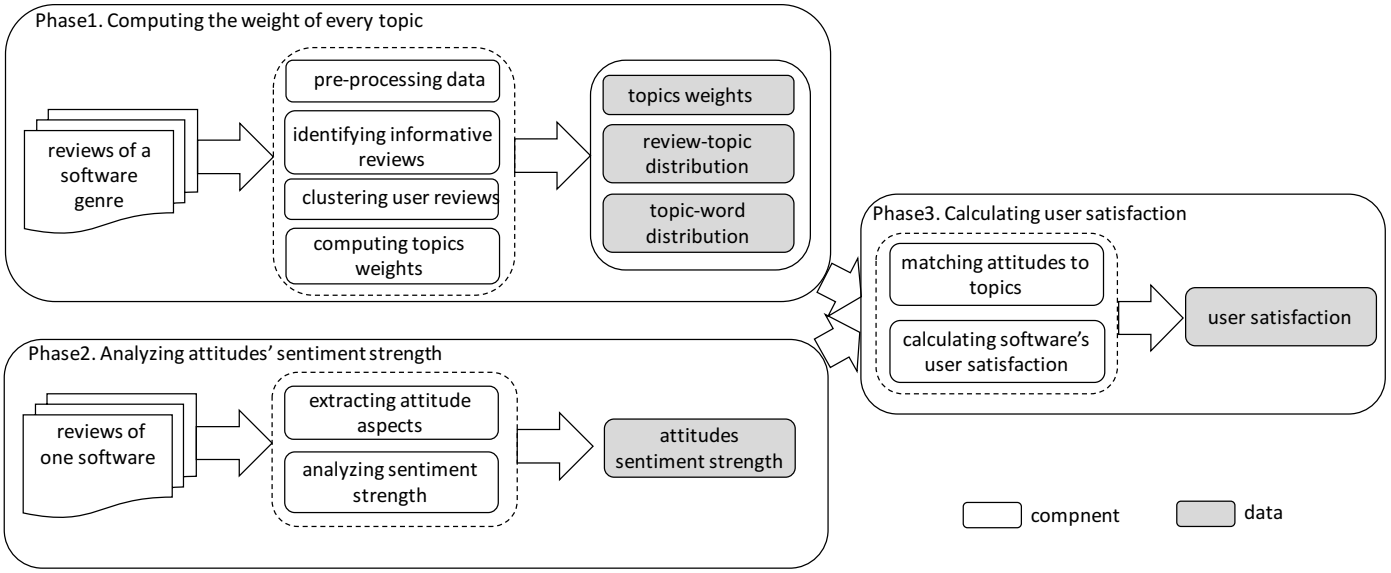

Fig. 1. Process of evaluating user satisfaction

Where W is words in the word map, w_i is the ith word in the word map, T is the number of topics, N is the number of words, $P(w_i|z_k)$ is the topic-word distribution calculated through LDA, $P(z_k)$ can be calculate through Eq.(2), $P(z_k|r_j)$ is the review-topic distribution calculated through LDA, and M is the number of reviews.

Computing topic weights. According to the results of LDA, we can get the review-topic distribution as TABLE I shown, z_k represents the kth topic and r_j represents the jth review. The weight of a topic is the accumulating of the probability of reviews belong to it, as Eq.(3) shown. For instance, if we would like to calculate the weight of topic z_1, we just accumulate 0.302, 0.001, 0.320, $\cdots$, 0.333, $\cdots$, 0.231.

$$Weight(z_k) = \sum_{j=1}^{M} P(z_k|r_j) \tag{3}$$

where z_k represents the kth topic, $P(z_k|r_j)$ is a probability among review-topic distribution calculated using LDA.

TABLE I. DISTRIBUTION FOR REVIEW-TOPIC

	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\cdots$	$\mathbf{z}_k$	$\cdots$	$\mathbf{z}_K$
$\mathbf{r}_1$	**0.302**	**0.000**	**0.004**	·	**0.033**	·	**0.209**
$\mathbf{r}_2$	**0.001**	**0.020**	**0.204**	·	**0.153**	·	**0.007**
$\mathbf{r}_3$	**0.320**	**0.000**	**0.004**	·	**0.013**	·	**0.331**
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
$\mathbf{r}_j$	**0.333**	**0..89**	**0.005**	·	**0.045**	·	**0.033**
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·
$\mathbf{r}_M$	**0.231**	**0.070**	**0.017**	·	**0.011**	·	**0.100**

C. Analyzing the Sentiment Strength of User Attitudes

In this section, we have two goals: find out attitude aspects and pure attitude reviews, and calculate their sentiment strength. *Sentiment Analysis* is used in this section to help us get the sentiment strength of attitudes. But it's different from traditional Sentiment Analysis which just analyzes the sentiment strength of sentence, SatisIndicator needs to analyze the sentiment strength of each attitude aspect in a particular sentence.

Extracting attitude aspects. We use a combination of methods of finding frequent nouns and noun phrases, and using opinion and target relation mentioned in [7] in this paper. First, nouns and noun phrases (or groups) were identified by a part-of-speech (POS) tagger and their occurrence frequencies are counted simultaneously. We set 5 as the frequency threshold, which means Nouns and noun phrases (or groups) appearing more than five times are attitude aspects. Then, we find sentences do not have a frequent aspect, but have some sentiment words, and extract the nearest noun or noun phrase to each sentiment word as attitude aspects (We assume that the same sentiment word can be used to describe or modify different aspects.).

Analyzing sentiment strength. As we can see in this software review:"*It has many really good functions, but it's ui is very very unfriendly.*". It is obvious that the user has positive sentiment on the attitude aspect "functions" and negative sentiment on the attitude aspect "ui". The sentiment strength of these two attitude aspects are also different, so we need to calculate sentiment strength of all attitude aspects in this review. We use an improved recursive neural tensor network (supervised method), which is proposed by Socher et al [13], to calculate sentiment strength of attitude aspect. The recursive neural tensor network was trained by the training set provided by the author[2]. The left graph in Fig.2 is Socher's method, which embeds every word in a sentence before giving it a score, and the sentiment strength of the sentence is on the root node. Our improved method shows in the right graph of Fig.2. First, we find out all of the subject, predicate, and object in a sentence. Then, we split them into different parts. Finally,

[2]http://www.socher.org/index.php/Main/HomePage#Publications

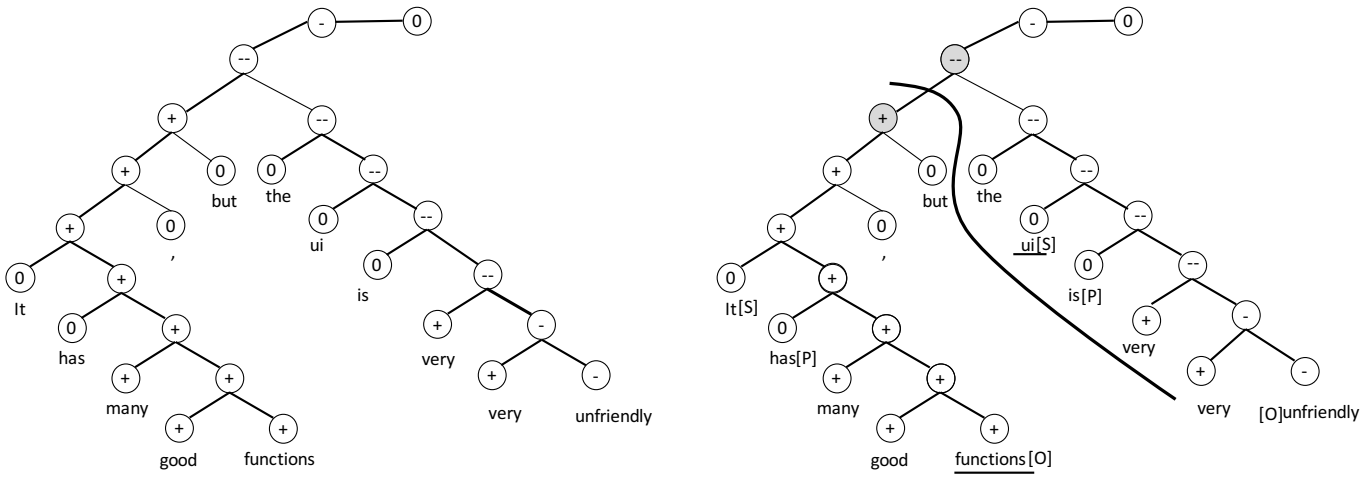

Fig. 2. Recursive Neural Tensor Network. The left one is proposed by Socher and the right one is modified by us to meet our demands.

we find the root node of these parts. If a part has an attitude aspect, the score at root node is its sentiment strength in this review. For example at the left graph in Fig.2 the curve split the whole sentence into two phases and the gray circles are the root node of these two phases. Then we can find attitude aspect in every phase. The attitude aspect "functions" gets a positive sentiment strength 4(+) and the attitude aspect "ui" gets a negative sentiment strength 1(–). Noted that if a review has no attitude aspect, we just calculate the sentiment strength of this review (the score at the root node of this review).

D. Calculating User Satisfaction of a Software

We have got the review-topic distribution $P(Z|R)$, the topic-word distribution $P(W|Z)$ and every topic weight in the first phase, and the attitudes sentiment strength in the second phase. In this phase, we will match the attitudes with topics, and calculate the user satisfaction of a software.

Matching attitudes to topics. For a specific attitude aspect or pure attitude review, we can find its topic use Algorithm 1. The inputs of Algorithm 1 are the review-topic distribution $P(Z|R)$, the topic-word distribution $P(W|Z)$ and the software reviews j'. The output is the topic the attitude belongs to. For s_{th} attitude aspect $a_{j',s}$ in a review $r_{j'}$, find the corresponding review-topic distribution $P(Z|r_{j'})$ and the corresponding topic-word distribution $P(a_{j',s}|Z)$, and then multiply them. The topic z_k which the attitude aspect $a_{j',s}$ belongs to can be taken when $P(z_k|r_{j'})$ times $P(a_{j',s}|z_k)$ is the max probability among others. For a pure attitude review, we compare the $P(Z|r_{j''})$. The topic z_k which the review $r_{j''}$ belongs to can be taken when $P(z_k|r_{j''})$ is the max probability among others.

Calculating user satisfaction. When we analyze the distribution of software review number, as Fig3 shown, we found that over 60% software only has one ore two informative reviews. Think about the situation as follow: we have two softwares "A" and "B", "A" has 50 positive reviews, and 50 negative reviews. "B" has 1 positive review and 1 negative review. All of them have 50% positive reviews and 50% of negative reviews. However, we can tell the confidence degree between these two softwares are different intuitively. We need to balance the proportion of positive ratings with the uncertainty of a small number of observations. Wilson Interval can be applied to correct the confidence degree and give a punish to the software with few reviews. The Eq.(4) is the Wilson interval proposed by Edwin B. Wilson [14], and Evan Miller applied it in voting system and got a significant

Algorithm 1 Algorithm of matching attitude and topic

Input:
review-topic distribution P(r_j|z_k), j∈ {1,...,M},
k∈ {1,...,K}
topic-word distribution P(w_i|z_k), i∈ {1,...,N},
k∈ {1,...,T}
Software Reviews $r_{j'}$ $j' \in \{1,...,M'\}$, $M' < M$

Output:
topic ID

for every review($r_{j'}$) in a software **do**
  if $r_{j'}$ has no attitude aspect **then**
    put $r_{j'}$ in pure attitude reviews;
  end if
  for every aspect($a_{j',s}$) in a review $r_{j'}$ **do**
    for every topic(z_k) **do**
      Probability←P(z_k|$r_{j'}$)*P($a_{j',s}$|z_k);
      topic←z_k;
      topic_Probability.put(topic,Probability);
    end for
    topicID($a_{j',s}$)← max(Probability(topic_Probability));
  end for
end for
for every $r_{j''}$ in pure attitude review **do**
  topicID($a_{j''}$)← max(Probability(P(z_k|$r_{j''}$));
end for
return $topicID$;

success [15]. It solves the accuracy problem of small samples very well. Use $\hat{p}$ as the number of positive attitude aspects add that of positive pure attitude reviews, n as the number of attitude aspects add that of pure attitude reviews, and z is a constant which indicates the statistic of a certain confidence level. In this paper we choose 95% as the confidence level. As we can see with the increasing of n (the number of attitude aspects add pure attitude reviews), the result will close to the $\hat{p}$. We multiply the user satisfaction by the center of Wilson Interval Eq.(5) to punish the software with few reviews.

$$WL = \frac{1}{1+\frac{1}{n}z^2}[\hat{p} + \frac{1}{2n}z^2 \pm z\sqrt{\frac{1}{n}(1-\hat{p}) + \frac{1}{4n^2}z^2}] \quad (4)$$

$$WL_{center} = \frac{\hat{p} + \frac{1}{2n}z^2}{1+\frac{1}{n}z^2} \quad (5)$$

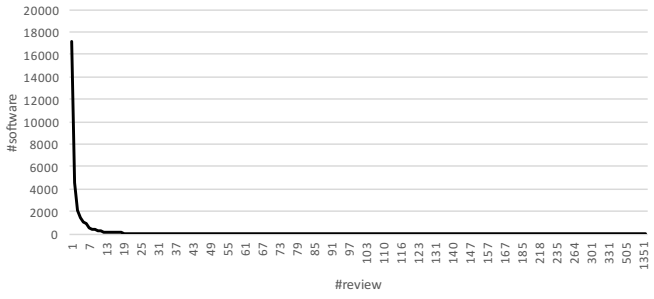

Fig. 3. Long tail phonomenan of software review number

We can calculate the user satisfaction of a software as the Eq.(6) shows.

$$Satisfaction = \frac{\sum_{j'=1}^{M'}\sum_{s=1}^{S} a_{j',s}W_{z_s} + \sum_{j''=1}^{M''} r_{j''}W_{z_{j''}}}{M'+M''} \cdot WL_{center} \quad (6)$$

Where $as_{j',s}$ represents the s_{th} attitude aspect sentiment strength in a review $r_{j'}$, z_s represents the topic which the attitude aspect $a_{j',s}$ belongs to. W_{z_s} represents the weight of topic z_s, M' represents the number of reviews which have attitude aspects, S represents the number of attitude aspects a review has, M'' represents the number of pure attitude reviews a software has and WL represents the center of Wilson Interval of this software.

III. Experiment

In order to evaluation if SatisIndicator has good performance in analyzing user satisfaction, we designed several experiments. We aim to answer the following questions: 1)Which model has better performance, IRNN (Improved Recursive Neural Tensor Network) or *SentiStrength*[3]. 2)What is the accuracy of SatisIndicator when used in analyzing user satisfaction. 3)What is the accuracy of SatisIndicator when used in analyzing user satisfaction of softwares with few reviews.

[3]http://sentistrength.wlv.ac.uk/

A. Data Preparation

In this section, we will show how to prepare the data which is used in the experiment. We wrote a crawler using jsoup[4] to get review data from Sourceforge.net. We have crawled all the ten software genres' reviews before Nov. 21, 2015. We only take *ffdshow* which is a software project of *Audio&Video* as an example to show how to calculate user satisfaction of a software.

First, we need to know the weight of every attitude topic. Since ffdshow is in Audio&Video genre, we analyze reviews in Audio&Video to get attitude topic weight. The *Audio&Video* software projects have 11580 software projects and 12480 reviews. Nature language precess(NLP) is used on those 12480 informative data which mentioned in *Pre-process data*. We randomly select 20% from this 12480 reviews to do manually annotate and use 5 people to label the 2496 reviews, if a review be labeled as informative more than three times, we annotate this review as informative, otherwise, annotate the review as non-informative. Classifier in lingpipe[5] is used to classify reviews that have been annotated, we found that Naive Bayes classifier based on 2gram has the best precision of 89.92%. So we use Naive Bayes classifier based on 2gram in lingpipe to classify all 12480 reviews. We got 12348 informative reviews totally. After that, we use these reviews as the input of LDA. We set $\alpha = \frac{50}{K}$, $\beta = 0.1$, and iterations $t = 1500$, use Eq.(1) to calculate the log-likelihood from topic number $k = 2$, to $k = 90$, we don't choose bigger topic number because the cure is decreasing and we don't think the topic number will bigger than 100 since we only have one domain. The result shows when $T = 25$, we got the max value of Eq.(1). In this case, we set 25 as the number of topics. When $k = 25$, we use JGibbLDA[6] to calculate review-topic distribution and topic-word distribution(the parameters are $\alpha = \frac{50}{K}$,$\beta = 0.1$,$K = 25$,$t = 1500$). Review-topic distribution can be used to compute the weight of every attitude weight.

Then, we need to calculate sentiment strength of each user attitude in ffdshow's reviews. We find both frequent and infrequent nouns and noun phrases like *Extracting attitude aspects* as attitude aspect. If a review have no attitude aspect, it is a pure attitude review. We use improved recursive neural tensor network to analyze the sentiment strength of every user attitude.

Finally, for every attitude aspect and pure attitude review we use Algorithm 1 to find its topic. We accumulate the ffdshow's attitude aspects sentiment strength multiply by their topic weights, and use Wilson Interval to punish the software with insufficient reviews as Eq.(6) shows.

For softwares in other 9 genres (Business & Enterprise, Communications, Development, Home & Education, Games, Graphics, Science & Engineering, Security & Utilities, System Administration), the user satisfaction can be calculated as the same.

[4]http://jsoup.org/

[5]http://alias-i.com/lingpipe/

[6]http://sourceforge.net/projects/jgibblda/

TABLE II. PERFORMANCE OF THE SENTISTRENGTH AND IRNN

	positive sentiment			negative sentiment			neutral sentiment		
	Precision	Recall	F_1	precision	Recall	F_1	Precision	Recall	F_1
SentiStrength	69.26%	78.83%	68.15%	70.32%	84.26%	76.61%	95.72%	43.16%	59.49%
Recursive Deep Model	90.28%	86.43%	84.83%	82.34%	85.47%	83.86%	75.64%	68.78%	72.05%

B. Model Selected for Sentiment Analysis

To our best knowledge, there is another work called sentiStrength also can be used to calculate the sentiment strength. We can input attitude aspects as keywords to assesses the sentiment strength of these keywords in the review or put a pure attitude review to assesses the sentiment strength of this review. We need to check which method has the better performance. We annotated 100 reviews manually for every genre in SourceForge.net.

The IRNN use ++(5), +(4), 0(3), -(2) and –(1) as the sentiment Strength of a word. The SentiStrength has different scoring criteria, it measures both the positive and negative sentiment strength of a sentence. For example, to the sentence *"It has many really good functions, but its ui is very very unfriendly"*, it will give the attitude aspect "functions" a sentiment strength set $[4, -1]$ which represents the positive sentiment strength is 4 and negative sentiment strength is -1, and the attitude aspect "ui" a sentiment strength set $[3, -4]$ which represents the positive sentiment strength is 3 and negative sentiment strength is -4. To make SentiStrength and IRNN can be evaluated in the same metric, we choose the bigger absolute values as the attitudes' sentiment strength (such as 4 for the "functions" and -4 for the "ui"). In the SentiStrength we define that the sentiment strength below 0 is the negative sentiment, the sentiment strength higher than 0 is the positive sentiment and the sentiment strength 0 is the neutral sentiment. In the IRNN we define that the sentiment strength below 3 is the negative sentiment, the sentiment strength higher than 3 is the positive sentiment and the sentiment strength 3 is the neutral sentiment.

We defined the precision, recall and F_1 as:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F_1 = \frac{2Precision \cdot Recall}{Precision + Recall} \tag{9}$$

TP means the user attitude should be in positive (negative or neutral) sentiment type and was correctly assigned to this type, FP means the attitude aspect should be in positive (negative or neutral) sentiment type but was assigned to negative or neutral (positive or neutral, positive or negative) sentiment type, and FN means that the attitude aspect should be positive (negative or neutral) sentiment type but be assigned to negative or neutral (positive or neutral, positive or negative) sentiment type. We use SentiStrength and IRNN to assess the positive sentiment, negative sentiment and neutral sentiment.

We run SentiStrength and IRNN on the 100 manually annotated reviews of each software genre in SourceForge.net. And the average precision, recall and F_1 of the ten software genre shows in TABLE II. It is obvious that IRNN works better in all of the sentiment type detection. Both models have high precision in positive sentiment. After analyzing the reviews, we found that because there are over 30% reviews simply write one or two positive sentiment words like *"Good"*, *"Good job!"*, and all these two models is based on supervised method, moreover *"good"* is a very common positive sentiment word in the training set, so both methods have a high precision and recall on detecting positive emotion. SentiStrength has high precision and low recall in neutral sentiment detection, because if it could not detect the sentiment type of the user attitude in a sentence, SentiStrength would classify it into neutral sentiment type, and give this user attitude 0 sentiment strength. Since we mainly concern about the positive and negative sentiment, we choose IRNN to analyze the attitudes sentiment strength.

C. Evaluating Model Performance

To answer the second question, we should choose the software with different number of reviews in each software genre averagely in SourceForge.net. But the number of reviews which a software are maldistribution. As Fig. 3 shown, there are over 60% software only has one or two reviews. So we first classify the software by review numbers, that means the software with the same review numbers should be in the same category, then we randomly choose one software in each N adjacent sub-topics. We totally choose 30 softwares form each software genre. We ask a volunteer to read all of the software reviews and give a satisfaction rank of all the 30 softwares in each software genre respectively. Here we choose to give them a rank sequence rather than a score, because satisfaction score is a subjective score which even we give a standard to the volunteer, it is still hard for him to provide a united score among softwares. However, the rank is relative, it is easier for a human to compare the satisfaction among softwares than give each software a satisfaction score. Since the volunteer will process so many softwares reviews, we choose the easier and quicker way which let the volunteer give out a rank sequence. We use two indicators to evaluate the accuracy of SatisIndicator, the first is precision@k and the second is Spearman Coefficient.

Precision@K. We want to know whether SatisIndicator can get a high average precision at the top 50% rank sequence, because users usually care more about softwares with high user satisfaction. So we use $p@K$ as Eq.(10) shown, to evaluate the average precision of the top 50% software in the rank sequence, and compared the results with the baselines of start rating (that means just use the star user given, for example 1 star, 2 star, etc.) without normalizing by Wilson Interval and star rating with normalizing by Wilson Interval (use Eq.(5) times the star rating, and $\hat{p}$ represents the number of reviews get 4 star or 5 star, n represents the number of reviews.).

$$p@k = \frac{|A \cap K|}{|K|} \tag{10}$$

A represents the set of the top K softwares in the sequence we calculated, K represents the set of top K softwares in

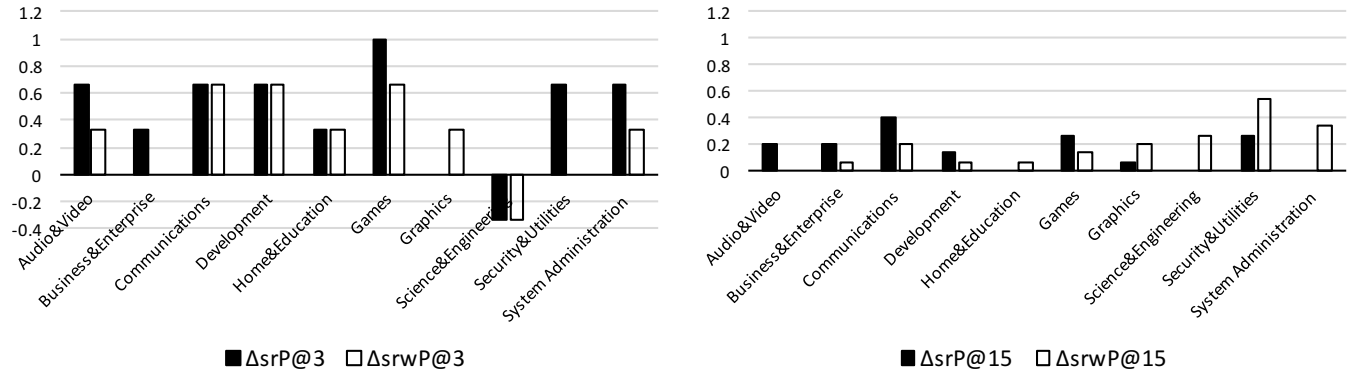

Fig. 4. Results of precision@3 and 15. The horizontal axis represents the software genre, the vertical axis represents the precision, ΔsrP@3 represents the precision of P@3 which SatisIndicator minus star rating baseline, ΔsrwP@3 represents the precision of P@3 which SatisIndicator minus star rating with Wilson Interval, ΔsrP@15 represents the precision of P@15 which SatisIndicator minus star rating baseline, ΔsrwP@15 represents the precision of P@15 which SatisIndicator minus star rating with Wilson interval.

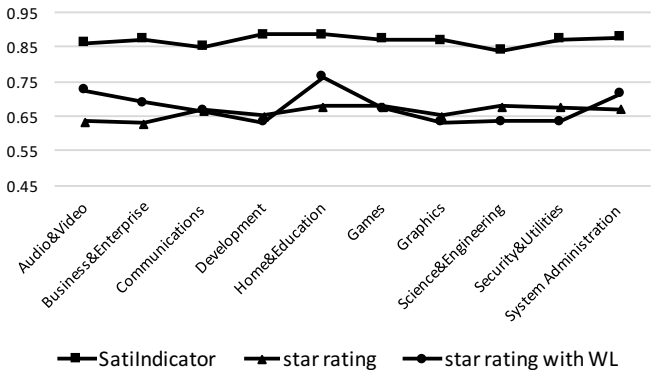

Fig. 5. Spearman Coefficient of SatisIndicator, star rating and star rating with Wilson Interval

the ground truth sequence, $|A \cap K|$ represents the number of softwares in the set A and set K at the same time and $|K|$ represents the number of softwares in the set K.

Some of the observations are in the Fig. 4. The black block means the $p@k$ of SatisIndicator minus the $p@k$ of the star rating baseline. The white block means the $p@k$ of SatisIndicator minus the $p@k$ of the star rating with Wilson Interval baseline. We found that SatisIndicator has good performance obviously at both $p@3$ and $p@15$.

The star rating baseline has a poor precision because it only has five level(1 star, 2 stars, 3 stars, 4 stars and 5 stars), and most softwares in the ground truth have five stars. So the star rating baseline cannot distinguish the difference between them, which makes the rank sequence random. The star rating with Wilson Interval baseline also has a poor precision. Because most reviews are positive, which make the $\hat{p}$ and n are very close to each other. And result the center of Wilson Interval is the same. But SatisIndicator gives a fine grain analysis to each word in a review and also give a punishment to a software with few reviews. So, even when the reviews are all have positive emotion we can still find the difference among them. The eighth ground truth (the Since&Engineering) has lower precision than both star rating baselines. We analyzed the data and find that there is only 3 software have 5 stars in the eight ground truth, so all this three softwares locate in the top 3 positions and make the precision of p@3 is 100%, but which is the first, which one is the second and which one is the third is random. But the sequence position among these three softwares is random. Although SatisIndicator only found 2 out of top 3 softwares, the two software's position is the same as the ground truth.

Spearman coefficient. To avoid the problem in the *Since&Engineering* data set in Fig. 4, which the sequence has high $p@3$ but low relevance to the ground truth, we use spearman coefficient, as Eq.(11) shown. It can evaluate the relevance among sequences.

$$\rho = 1 - \frac{6 \sum_{i=1}^{N} (s_{1,i} - s_{2,i})^2}{N(N^2 - 1)} \tag{11}$$

the $s_{1,i}$ represents the i_{th} position in sequence 1, the $s_{2,i}$ represents the i_{th} position in sequence 2. N represents the number of position both sequences have. The Spearman Coefficient is between -1 to 1. If the Spearman Coefficient close to 1, that means the two sequences are very similar. If the Spearman Coefficient close to 0, that means the two sequences are very different. And if the Spearman Coefficient close to -1, that means the two sequences are negative correlation.

Some of the observations are showed in the Fig. 5. It is obvious that SatiIndictor has very high Spearman Coefficient over 80% on all of the ten data sets. That means the sequence calculated by SatisIndicator is more relevance to the ground truth.

D. Evaluating Model Performance when Reviews are Insufficient

When analysis the user reviews in SourceForge, we found that over 95% softwares with one or two reviews have five stars. In such situation, star rating can only give a random sequence. Furthermore, we also found that about 67% softwares

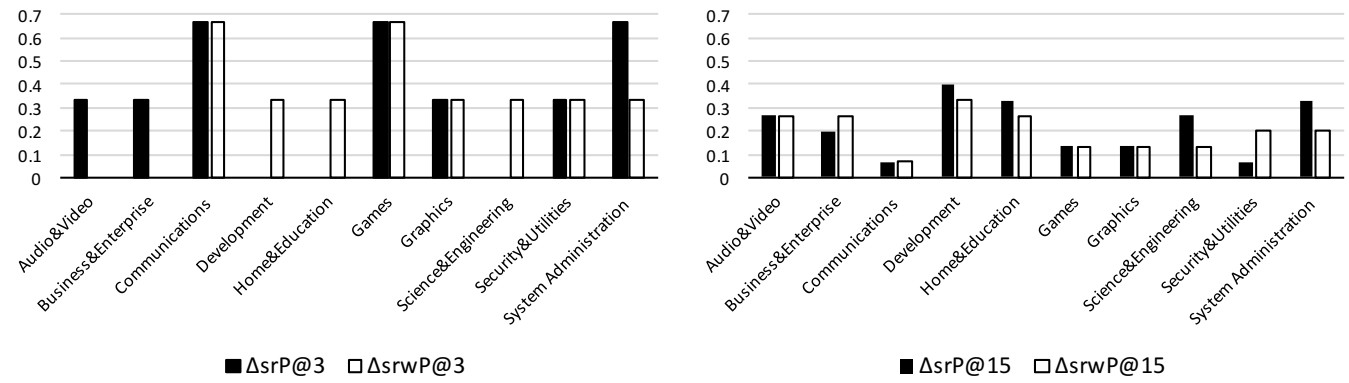

Fig. 6. Results of precision@3 and 15 when reviews are insufficient. the horizontal axis represents the software genre, the vertical axis represents the precision, ΔsrP@3 represents the precision of P@3 which SatisIndicator minus star rating baseline, ΔsrwP@3 represents the precision of P@3 which SatisIndicator minus star rating with Wilson Interval, ΔsrP@15 represents the precision of P@15 which SatisIndicator minus star rating baseline, ΔsrwP@15 represents the precision of P@15 which SatisIndicator minus star rating with Wilson Interval.

have one or two reviews in SourceForge. It is meaningful to know if SatisIndicator performance well when the softwares have few reviews. We choose a sample of 30 softwares with least reviews in every software genre. It is very hard even for human to give the sequence only according to one or two reviews. So we ask the volunteer to choose softwares which have the least reviews but can give a sequence use their reviews. The software we choose has 1 to 7 reviews in all software genres.

Precision@K. We found that StatiIndicator obviously has good performance at $p@3$ and $p@15$ over the other two baselines when softwares have few reviews. The star rating has a very bad performance at $p@3$ and $p@15$. Because most software only has one review, and we find that almost every software which has one review has five stars. Almost all softwares in the ground truth have the same star rating, and the sequence calculated using star rating makes no sense. The star rating with Wilson Interval also performances bad. Because as Eq.(5) shown, when all softwares with one positive review will have the same center of Wilson Interval (the positive reviews $\hat{p}$ equals the number of reviews n). When evaluate the software with one positive review, the result will as same as the star rating without Wilson Interval. SatisIndicator has better performance because we have analyzed every attitude aspects in a review sentence. Although the software has only one review, it still has different attitude aspects. These attitude aspects may in different topics, that means attitude aspects will have different weight. Different from star rating with Wilson Interval the $\hat{p}$ in Wilson Interval of SatisIndicator represents the positive attitude aspects numbers, so the center of Wilson Interval will be different. With such fine grained, the result of SatisIndicator will more similar to the ground truth. However, if the software only with one review like *"Good!"*, *"Great job!"*, SatisIndicator will almost have the same result as the two baselines. But in this situation even human can not tell which software has a higher user satisfaction from reviews.

Spearman Coefficient. Some of the observations are shown in the Fig. 7. It is obvious that SatisIndicator has a very high Spearman Coefficient over 60% on all of the ten data sets when softwares with few reviews. That means the sequence

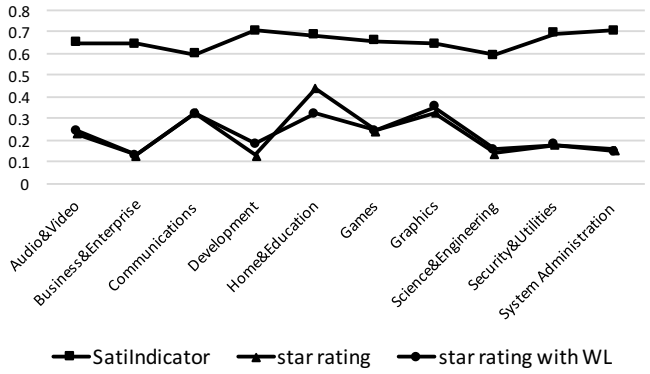

Fig. 7. Spearman Coefficient of SatisIndicator, star rating, and star rating with Wilson Interval.

calculated by SatisIndicator is more relevance to the ground truth, even when all of the software has few reviews. The other two baselines have almost random sequence. Because they only give a coarse grained score to a software review, not a fine grained score to every attitude. And when all of the softwares with few reviews have 5 stars, the sequence is random. When all of softwares have few reviews and all of them are positive reviews with 5 star, the both two baselines can not tell which software has high user satisfaction and which software has low user satisfaction.

IV. RELATED WORK

Our work is related to sentiment analysis, and user reviews mining.

A. Sentiment Analysis

Sentiment analysis, also known as opinion mining, is the extraction of positive, negative or neutral sentiment from text[5]. Academic community has done many work on this topic, also have made great contributions, but we can't use those outcomes directly.

Pang et al.[6] use supervised learning and classify reviews by overall sentiment. We can not use their work since they just classify reviews into three topic: positive, negative, and neutral, and they just calculate the review's sentiment not every attitude aspect's.

Hu et al.[7] use frequent features and infrequent features as product features commented by customers, then they identifying opinion sentences in each review, and deciding whether each opinion sentence is positive or negative, finally they can get the result that for a feature, there are how many sentences express positive emotion and how many sentences express negative emotion.

Our work is different from their because we want to get the features'/aspects' sentiment strength rather than how many positive or negative reviews these features/aspects have. So, we can't use those work to do a fine grained quantitative calculation on sentiment strength of user reviews.

Popescu et al.[8] decompose the problem of review mining into the sub-tasks of identify product features, identify opinions regarding product features, determine the polarity of opinions, and rank opinions based on their strength(for example, *"horrible"* is a stronger indictment than *"bad"*). But Still can't quantitatively evaluate the sentiment strength of the features.

Mike et al.[9], extracted sentiment strength from informal English text, using new methods to exploit the defacto grammars and spelling styles of cyberspace. Applied to MySpace comments and with a lookup table of term sentiment strengths optimized by machine learning and make a tool called SentiStrength. SentiStrength can evaluate sentiment strength at a level of positive 1 to 5 and negative -1 to -5. and 0 represents neutral. We can't directly use their work to evaluate the Satisfaction, because they extract attitude aspects without categorising them, and we need to find not only the features' sentiment strength but also the weight of every feature.

B. Mining User Reviews

There are several studies on user reviews mining recent years, but most of them are helping developers to quickly read the user reviews, or help developers to quickly understand the user experiments from the user reviews.

Galvis et al. [16] do requirements analysis based on user reviews. They clustering user reviews by topic model, then get requirements report for developers by quality indicators which generated by clustering the user reviews by topic model. This will help developers save a lot of time to read user reviews, but it helps nothing for the software quality evaluation. We not only need know what user talk about or care about, but also the user attitudes toward them. That means we should analysis the sentiment even the sentiment strength of the reviews. their work is not enough for our demands.

Chen et al. [17] propose an approach to rank user reviews by importance for developer, to help developers find out informative reviews. They just cluster user reviews and tell the developer which topic is important. We can not use their approach since it either analyze the sentiment of the user reviews or can deal with software with insufficient reviews.

Syn et al. [18] propose an approach to category user reviews according to the three characteristics in quality-in-use model defined in ISO: effectiveness, efficiency and freedom from risk. We can't use their approach because we evaluate another characteristics of quality-in-use model: satisfaction.

Leopairote et al. [19] propose an approach for software product reviews mining based on software quality ontology constructed from ISO 9126 and a rule-based classification to finally produce software quality in use scores for software product representation. We can't use their approach since they have only calculated the review numbers of positive, negative, and neutral, and do not calculated the strength of the sentiment.

Maalej et al. [20] classify user reviews into four topics: bug report, feature request, user experiences, and rating. We can't use this method because we don't know how many topics that user reviews contain, that means we should use cluster not classify to find how many topics user reviews contain.

What Guzman et al. [21] did is similar with us, their work aim to help developers filter, aggregate, and analysis user reviews by topic model. But they use topic model on single software, since topic model is only suit to deal with large number of reviews, that means 99% percent of software project on SourceForge.net can't be used their approach. Furthermore their work just analyze the sentiment strength of a review, not every attitude aspects.

V. Conclusion

In this paper, we proposed a novel approach called *SatisIndicator* to automatically evaluate the user satisfaction from user reviews which implicate user attitudes. User reviews contain many user attitudes to softwares, which can be using to evaluate the user satisfaction. First We use topic model to cluster reviews, and weight each topic. Then we extract user attitudes, and use an improved recursive neural tensor network to analyze attitudes sentiment strength. Finally we get the software user satisfaction through multiplying attitudes sentiment strength by their corresponding topic weight. Wilson interval is used to punish softwares with few reviews. We want to use quality-in-use model to assess the quality of software projects automatically in the future. However, user satisfaction is one of characteristic in quality-in-use model, there are also other important characteristics like effectiveness, efficiency, freedom from risk, and context coverage. All characteristics need to be assessed using both subjective and objective data. We also want to map the quality-in-use model with software project life cycle to assess software project quality in different life cycle stages in the future.

Acknowledgment

This research is supported by 973 Program in China (Grant No. 2015CB352203) , National Natural Science Foundation of China (Grant No. 61472242), and Key Lab of Information Network Security, Ministry of Public Security (Grant No. C14609).

REFERENCES

[1] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, K.-i. Matsumoto, B. Ghotra, Y. Kamei, B. Adams, R. Morales, F. Khomh, *et al.*, "The impact of mislabelling on the performance and interpretation of defect prediction models," in *Proc. of the 37th Int'l Conf. on Software Engineering (ICSE)*.

[2] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *Proc. of the 37th Intl Conf. on Software Engineering (ICSE)*, 2015.

[3] X.-Y. Jing, Z.-W. Zhang, S. Ying, F. Wang, and Y.-P. Zhu, "Software defect prediction based on collaborative representation classification," in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 632–633, ACM, 2014.

[4] "Iso/iec 25010:2011: Systems and software engineering systems and software quality requirements and evaluation (square) system and software quality models (2011),"

[5] 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.

[6] 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*, pp. 79–86, Association for Computational Linguistics, 2002.

[7] 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*, pp. 168–177, ACM, 2004.

[8] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Natural language processing and text mining*, pp. 9–28, Springer, 2007.

[9] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[10] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, ACM, 1999.

[11] 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.

[12] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[13] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, p. 1642, Citeseer, 2013.

[14] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927.

[15] E. Miller, "How not to sort by average rating," *URL: http://www. evajimiller. org/how-not-to-sort-by-average-rating. html*, 2009.

[16] 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*, pp. 582–591, IEEE Press, 2013.

[17] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 767–778, ACM, 2014.

[18] W. T. W. Syn, B. C. How, and I. Atoum, "Using latent semantic analysis to identify quality in use (qu) indicators from user reviews," *arXiv preprint arXiv:1503.07294*, 2015.

[19] W. Leopairote, A. Surarerks, and N. Prompoon, "Software quality in use characteristic mining from customer reviews," in *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, pp. 434–439, IEEE, 2012.

[20] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pp. 116–125, IEEE, 2015.

[21] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pp. 153–162, IEEE, 2014.