2016 12th International Conference on Signal-Image Technology & Internet-Based Systems

Stratify Mobile App Reviews: E-LDA Model Based on Hot "Entity" Discovery

Yuandong Liu, Yanwei Li,
BUPT National Engineering Lab for Mobile Network Technologies
National Computer Network Emergency Response Technical Team/Coordination Center of China
Beijing, China
cyrus.cl@outlook.com

Yanhui Guo, Miao Zhang
BUPT National Engineering Lab for Mobile Network Technologies
Beijing, China
{yhguo, zhangmiao}@bupt.edu.cn

***Abstract*—Recent literatures have illustrated approaches that can automatically extract informative content from noisy mobile app reviews, however the key information such as feature requests, bug reports etc., retrieved by these methods are still mixed and what users really care about the app remains unknown to developers. In this paper we propose a novel model SAR: Stratify App Reviews, providing developers information about users' real reaction toward apps. SAR stratifies informative reviews into different layers, grouping the reviews based on what users concern, and we also develop a method to compute the user general sentiment on each entity. The model performs user-oriented analytics from raw reviews by (i) first extracting entities from each review, identifying hot entities of the app that users mostly care about, (ii) then stratifying all the reviews into different layers according to hot entities with a four-layer Bayes probability method, (iii) and finally computing user sentiments on hot entities. We conduct experiments on three genres of apps i.e. Games, Social, and Media, the result shows that SAR could identify different hot entities with respect to the specific categories of apps, and accordingly, it can stratify relevant reviews into different layers, the sentiment value of each entity can also represent users' satisfaction well, we also compared the result with human analysis, with the similar accuracy, the SAR can speed up the overall analysis automatically. Our model can help developers quickly understand what entities of the app users mostly care about, and how do they react to these entities.**

***Index Terms*—App Review, Bayes Probability, Sentiment Computation, Entity.**

I. Introduction

Recently, with the development of mobile Internet, mobile apps are thriving at large [16], which connect to everyone's life. Users download apps from app stores, and write reviews to share their experience about the app performance, these reviews contain valuable information for app developers, also attract to latent users [2]. Therefore, finding valuable information from these noisy reviews is of importance for both developers and users. As apps updated periodically and new app released, the app reviews increased tremendously everyday [33]. In this case, human reading of these apps reviews would be tedious and time consuming.

In order to improve users experience, reading app reviews is an effective way for developers to understand what users think about the app and what do they really need [14,17,18]. Recent works have proposed ways to extract informative content from raw reviews [3,12,39], such as extracting feature requests through linguistic rules [1], classifying reviews into different categories, also some works conduct sentiment analysis on these reviews to compute user's satisfaction towards the app [5]. Some literatures also develop tools to help developers and users to find different types of reviews [35]. All these works have been recently conducted due to the high frequency of app using in daily life, applying nature language processing and machine learning algorithm. Previous works on app views mining mainly focus on the linear level of Natural Language Processing [38], mostly they concentrate on the key information extraction, filter the noisy and useless reviews, or using NLP technology divide them into different topics. When we look into different categories of apps such as Games, Education, and Media etc., those apps have quite different functions, and users focus on totally different aspects when writing reviews. The previous work [25] on these reviews can extract all the feature requests and classify them, but no further works have been conducted. Therefore, identifying hot entities most concerned by users toward different apps, and grouping the related reviews into fine grained topics would be an effective way for developers to understand users and to improve their apps. However, we find no research work in this perspective has been done, whereas a fine grained sentiment analysis on app reviews have been conducted [11], and also a tool developed for app reviews mining based on key words [37].

In this paper, we conduct a survey about the user preference among different categories of apps they use. We find out that users are more focused on the scenic design, graphics and sound effect of a game app, whereas for a social app, these would switch to sharing, loading, notification etc., also with respect to other type of apps, it would be other aspects. We denote these aspects as entities, there are a bunch of reviews written by users related to each entity. Entities would have the same type of bug reports or features requests, or function problem etc., for example, reviews from a social app like Facebook, with respect to UI design and Functions, users may have hundreds of reviews towards these two entities, which both have feature requests and bug reports, and details like remove or add the specific items. In this case, the recent work [21] extracts all reviews containing such information but still a bit noisy to figure out which entities users want to change most, and how users react to these entities remains unknown. Considering these two situations, we term the problem Stratify App Reviews, and based on the previous research we propose the solution model SAR.

The model consists of three parts: hot entity discovery, stratifying reviews and entity sentiment analysis, the structure of SAR is showed in Fig. 1. Here we denote entity as the

978-1-5090-5698-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SITIS.2016.97

combination of nouns that appear frequently in reviews, E-LDA is our modified topic model based on LDA model [4]. The three parts of our model we proposed can help developers quickly notice the most user-concerned entities, and go into the hierarchical group of reviews for the details, therefore developers can understand which aspects of the app should be improved and enhanced, as well as which parts that users are not fond of. With the sentimental value of each entity, developers can intuitively observe how users react to the app design.

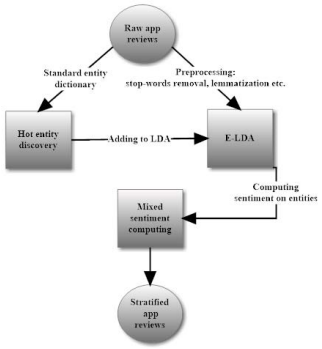

Fig. 1. Overview of SAR

The remainder of this paper is structured as follows. Section II introduces hot entity discovery method. Section III describes stratified app reviews mining approach. Section IV presents the sentiment computing of each entity. Section V shows our empirical evaluation of SAR and case studies. Finally, related works are discussed in Section VI and we conclude our study in Section VII.

II. Hot Entity Discovery

Generally, users writing reviews about different traits of an entity, the first step of SAR model is finding out those entities that mostly reviewed by users. The entity extraction from text set is the mainly studied filed in nature language process. We denote R as a set of all reviews from one specific app, using the *Stanford Named Entity Recognizer*[1], we obtained all entities from R. Since most users are not professional in mobile app develop, when they writing reviews about one function or feature of the app, they might use different words to describe it, e.g., user would write picture review, photo download, and picture deleting about one social application, however in developer's perspective, and these are relevant to one entity that is photo processing.

In order to make entities more representative and effectively describe the general aspects of an app design, we build a standard app develop entity dictionary based on the developer document [2]. The dictionary records standard specific components that a developer need to release a high quality app. We define all the data sets as follow:

$$\begin{aligned} E_s &= \{ \text{ the standard developer entities } \} \\ E_r &= \{ \text{ entities of a review from R } \} \\ E_{hot} &= E_k \cap E_s = \{ \text{ Hot entitis of an app } \} \\ \cap &\text{ denote the Word2Vec relation computing} \end{aligned} \quad (1)$$

The hot entity is the one that mostly concerned by users, this means high frequency of appearance in E_r, we use Word2Vec, a distributed, vector-based representation of words [40], to compute the relationship of entities inside the dataset E_r, as well as entities between E_s and E_r. By doing this, we are able to count the frequency of hot entity mentioned in (1), rank them and extract hot entities that have high value of relation with entity in E_s. The entity discovery algorithm can be described in table I.

For the entity that have high frequency in E_r but low relation value, the algorithm would directly put it into E_{hot}.

III. Stratify Reviews

To stratify the reviews that are relevant to the entities found above, we modified the existing topic model Latent Dirichlet Allocation (LDA) [4], add an entity layer, we denote the model as Entity-LDA (E-LDA). In this approach, we are able to classify the reviews into different topics according to different entities, classifying all app reviews into topics according to entities that users mostly care about. Since the hot entity discovery algorithm would find entities that are mostly talked about by users in a specific period, E-LDA can help developers quickly figure out what need to be improved or modified about the app, which make developers much more competitive in the mobile marketplace.

TABLE I
ENTITY DISCOVERY ALGORITHM

```
Input(R)
Get entity set: Er, Es, Ehot
num[]:frequency of an entity
k=sum(E_r), s=sum(Es)
for i=1,2,...,k do
  for j=i+1,...,k do
    value[i,j] = Word2Vec(Er[i],Er[j])
    if value[i,j]>0.5
      denote Er[i]=Er[j] as the same entity
       num[i]+1
      end
Each Er[] in top Q num[] entities:
  for i=1,...,Q j=1,...,s do
    if Word2Vec(Er[i],Es[])=0
      Ehot=Er[;]
    else Ehot=Es[j] s.t. Word2Vec(Er[i],Es[j])=maxValue
export Ehot
```

A. E-LDA Topic Model

The topic model LDA is a probabilistic distribution algorithm which uses Gibbs sampling to assign topics to documents, and each topic is a probabilistic distribution over words, thus each document is modeled as a mixture of topics. It is efficient to analysis the short message like twitter, microblog, in our case app reviews. The model consists of three layers: documents, words, topics, i.e. $\vec{\varphi}_k$ denotes the k^{th} topic to words distribution and $\vec{\vartheta}_m$ denotes the m^{th} document to topics distribution:

[1] http://stanfordnlp.github.io/CoreNLP/ner.html

[2] https://developer.android.com/guide/index.html

$$\Theta = \{\ \vec{\vartheta}_m \sim \mathrm{Dir}(\vec{\alpha})\ \}_{m=1}^{M}\ \ M\ \ documents$$
$$\Phi = \{\ \vec{\varphi}_k \sim \mathrm{Dir}(\vec{\beta})\ \}_{k=1}^{K}\ \ \mathrm{K}\ \ topics \tag{2}$$
$$Dir(\vec{\rho}\,|\,\vec{\beta}) \triangleq \frac{\Gamma(\sum_{k=1}^{K}\beta_k)}{\prod_{k=1}^{K}\Gamma(\beta_k)}\prod_{k=1}^{K}\rho_k^{\beta_k-1}\ Dirichlet\ \ allocation$$

Fang LI et al. [29] extends the LDA model by adding a tag layer between the document and topic layer, to effectively capture semantic knowledge from blogs, and gives the parameter estimation method. In this paper, based on Fang's work, we add an entity layer to the original LDA and propose a E-LDA model to stratify app reviews. Our approach can be described in Fig. 2.

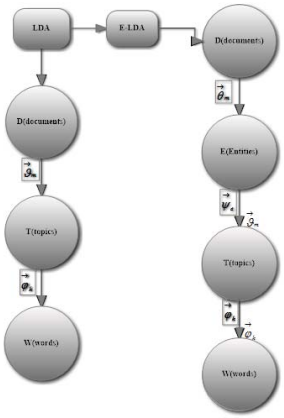

Fig. 2. Adding entity layer to LDA

By adding an entity layer， the document layer and topic layer, the distribution formula in (2) is modified in (3):

$$\Theta = \{\ \vec{\theta}_m \sim \mathrm{Dir}(\vec{\eta})\ \}_{m=1}^{M}\ \ M\ \ documents$$
$$\omega = \{\ \vec{\psi}_e \sim \mathrm{Dir}(\vec{\beta})\ \}_{e=1}^{E}\ \ \mathrm{E}\ \ entities \tag{3}$$
$$\Phi = \{\ \vec{\varphi}_k \sim \mathrm{Dir}(\vec{\alpha})\ \}_{k=1}^{K}\ \ \mathrm{K}\ \ topics$$
$$Dir(\vec{\rho}\,|\,\vec{\alpha}) \triangleq \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\rho_k^{\alpha_k-1}\ Dirichlet\ \ allocation$$

Using this approach, we can get three distribution matrices, document to entity matrix Θ, entity to topic matrix ω, topic to word distribution matrix ϕ. Then we extend the LDA to a four layer Bayes probability model. Therefore, the E-LDA model is illustrated in Fig. 3.

For each word *w* in a document *m*, an entity *e* is sampled from the entity distribution θ_m, then a topic *t* is drawn based on entity *e* from the distribution ψ_e, following, the word *w* is drawn based on the topic *t* from distribution φ_k. The document *m* is generated by repeating the process N_m times, which is the number of word tokens in document *m*.

B. App Reviews in Stratified Structure

Using E-LDA model, we can obtain the documents to entities distribution, entities to topics distribution, and topics to words distribution, thus we can get topics that are relevant to each entity according to entities to topics distribution, therefore, we can stratify reviews into entity-topic-document structure. If the probability of a topic in an entity exceeded the threshold, we

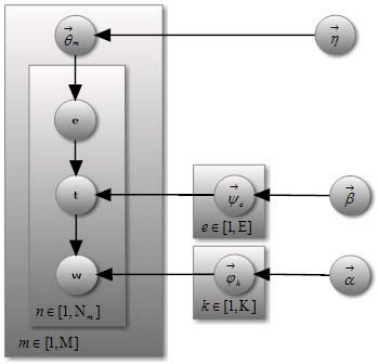

Fig. 3. Entity-LDA model

believe that the topic and entity are relevant, otherwise they are irrelevant. For each entity, we can select the relevant topics according to matrix ω with a threshold, then user reviews can be grouped into the topics according to matrix ϕ with a threshold, therefore all app reviews will be stratified into a three-layer structure showed in table II.

IV. Entity Sentiment Computation

Normally, for one of entities we analysis above, when developers start to design, it consists of many aspects, e.g., an

TABLE II
APP REVIEWS STRATIFIED STRUCTURE

- Entity 1:
  - Topic 1:
    - Review 1
    - Review 2
    - …
    - Review N
  - Topic 2:
    - Review 1
    - …
    - Review N
  - …
  - Topic K
- Entity 2:
  - …

entity like UI design[3], it would include the layout, color, and front etc. In general, we find out that people might be annoyed by one specific trait of an entity, but they are quite pleased by the rest part of entity, therefore they usually come up the problem in their reviews and instead give a high rating. In this situation, the rating is positive, however the reviews may tend to be negative, for some others the case would be vice versa.

A. Problem Summary

To some extent, the ratings relate to each review represent the user general attitude toward the app, positive, negative, or neutral, however, we find some users may give a high rating even they have complained about one specific function in reviews, e.g., the review from a widows store app Microsoft

[3] https://developer.android.com/guide/topics/ui/index.html

Math with a rating 4 (scale 1-5):

There seems to be some mistakes within the exercises, like the solution explanation shows the sign with a + whereas the actual problem has a - or instead of a + sign there is a multiplication sign. But all in all the app is good and works well.

Developers can directly understand that users are satisfy with the app from ratings and it is highly possible for them to ignore the details of the reviews with high ratings, which actually contain the valuable information about app improving given by users, in this case, the rating is not well presenting the users' sentiment on apps, and it can also blind developers. Therefore, a new sentiment calculation method which takes into consideration both user ratings and reviews, need to be proposed.

B. Mixed Sentiment Computing Solution

SentiStrength [9] is a lexical sentiment extraction method specialized in analysis on short, low quality text. Based on the idea that humans can express both positive and negative sentiments in the same sentence, SentiStrength assigns positive scores in the [+1, +5] range, where +5 denotes an extremely positive sentiment and +1 denotes the absence of sentiment. Similarly, the negative sentiments range from [-1, -5], where -5 denotes the extremely negative sentiment and -1 indicates the absence of any negative sentiment. For most ratings showed in app stores like Google Play, Apple Store, and Windows Store, the rating is a positive number ranges from [0, 5].

The algorithm we proposed is based on the SentiStrength, which calculates user sentiment on app entities, combing with ratings and the text that users give. This algorithm can help developers intuitively know what users mostly like and what need to be improved through sentiment values. For each review which relate to an entity using E-LDA, we record its ratings at the same time, H denotes the combination value of entity ratings R_e and SentiStrength of reviews S_r. R_e is the average of ratings that relate to the entity, we set a default value 0 where a review without rating. Similarly, S_r is the average of general sentiment on reviews of the entity. To combine this two parts, firstly we calculate the percentage G_0, which denotes the ratio of R_e to its full rating scale, P_0 and Q_0 represent the positive value and the absolute value of negative sentiment respectively in S_r. Then we modify P_0 and Q_0 according to G_0, the algorithm can be described in TABLE III, N_0 denotes the scale of modified SentiStrength.

In this method, we assign H a new scale N=5, which makes H ranges from [-5, 5], where -5 denotes an extremely negative sentiment, whereas +5 means the extremely positive sentiment.

TABLE III
MIXED SENTIMENT COMPUTING ALGORITHM

```
Define: N is new scale H[-N, N]
 temp=abs(G0-0.5)
 N0=5+(1-0.5)*5=7.5
 case G0<0.5:
    P1=P0, Q1=Q0 + N0*temp
 case G0>0.5:
    P1=P0 + N0*temp, Q1=Q0
 General percentage G=p1/N0 - Q1/N0
 H=Ave(G*N)
Output H
```

[4] https://github.com/MarcelloLins/GooglePlayAppsCrawler

V. EMPIRICAL EVALUATION

We study the apps uploaded by developers in mobile app stores like Google Play, Windows Store, and Apple Store. Each store has their user feedback session, allowing users give feedback, although rules of these mobile stores are slightly different in some ways, they all include app reviews and ratings. We develop a tool based on the project online[4], by using it, we scrawl six apps of three categories from US app store Google Play, our SAR model is estimated by using these data.

A. Entity Discovery for A Specific App

We fetch six apps data which are from game, social, and media categories respectively, each category has two apps to be analyzed. We choose apps from different categories because we want to evaluate the hot entity discovery part of SAR model, to test if it could discovery different entities in different app genres. For every two apps from the same category, we want to measure that if our method could discovery the app hot entities that are special to the app design and user experience, even in the same category, they may have common entities, whereas two different apps would focus on different theme, therefore the entities would be different. The overview of these app data is shown in TABLE IV.

We implement our entity discovery algorithm to these app data analysis, the result is shown in TABLE V. For apps in different categories, the entities discovered by our model are distinctive, and present the common traits of the category, e.g., for the game app *Bubble Shooter*, since it is a game app, what users write most about the app is *Advertise* and its game rules *Levels*, as well as the *Graphics* etc., whereas for the media app *Speaker Boost,* which the entities are *Speaker*, *Volume* etc. This indicate that users download this app mostly for its practical

TABLE IV
OVERVIEW OF THE EVALUATION APP DATA

App	Category	#Reviews	#Ratings	ΦLength
Virtual Table Tennis	Game	285,091	285,086	86
Bubble Shooter	Game	54,461	54,461	53
Quora	Social	164,002	164,002	125
Tumblr	Social	2,177,329	2,018,029	65
YouTube	Media	11,646,297	11,645,981	72
Speaker Boost	Media	29,631	29,631	56

TABLE V
APP HOT ENTITY DISCOVERY

App	Hot entities
Virtual Table Tennis	Shot, Advertise, Data, Spin Meter, Paddle, Online Playing
Bubble Shooter	Advertise, Levels, Bubble Shot, Challenging, Gift Box, Graphics
Quora	Question Answer, Not Working, Content, Bug Fix, Images, Website
Tumblr	Blog Post, Dash, Tags, Website Browser, Profile Customization, Video Gifts
YouTube	Subscription, Livestream, UI, Video Quality, Advertise, Playlist
Speaker Boost	Sound Cloud, Toggle, Speaker, Headphone, Volume, Podcast

functions that are quite different from the game genre. This result shows that our hot entity algorithm can extract the entities that users mostly concerned about, regardless of the app categories. On the other hand, inside the same category, the algorithm can also find the common entity between two different apps. For example, the *Virtual Table Tennis* and *Bubble Shooter* are game apps, and our algorithm finds the common frequently mentioned entity *Advertise*, this indicates that, firstly, our method have a high efficiency on hot entity discovery of app reviews; secondly, our algorithm can find a trait of the app category, e.g. the game genre that the advertisement is the common problem users complain about. For the rest part, we implement our E-LDA model to stratify these six apps according to the hot entities discovered in TABLE V.

TABLE VI
E-LDA DISTRIBUTION RESULE OF BUBBLE SHOOTER REVIEWS

Entity	High probability topic	High probability reviews
Levels	topic11(P=0.3634)	All levels are pretty easy. Wish the levels got a little harder as you move up. But overall a good game.
	topic8(P=0.2356)	Its like totally awesome. I play every day im up to level 658 lolsz
Challenging	topic15(P=0.4652)	I really like this game but it is just a little too easy if this was a little more challenging I might give it more stars.there are also a couple adds that were annoying but other than that this is a great game ;)
	topic6(P=0.3221)	Fun and challenging but the most frustrating challenge is having to close pop up ads every 20 to 60 seconds of game play. Life is too short for such BS.
Advertise	topic10(P=0.5783)	The amount of ads is to much. Every completed level and you get an ad. Theyre easy so its ad after ad after ad. Ill look for another game im sure ill find one similar
	topic4(P=0.4183)	Addictive game a chalange for free time.ads is a problem but i hope it will update with add free version.

B. SAR Application

We use reviews from Google Play[5] to develop and evaluate our E-LDA model, however, the model can also be applied to reviews from other platforms. After we gathering the data and extracting entities form each app, a preprocessing of raw app reviews is conducted. The preprocessing of the reviews involves stop-words removal, lemmatization, and nouns, verbs, adjectives extraction by using *Stanford CoreNLP*[6]. The TABLE VI shows the result of E-LDA model applied to game app *Bubble Shooter* reviews. For an entity Levels, the topic11 and topic 8 have the highest probability 0.3634 and 0.2356 respectively, this indicate that the two topics are related to the entity *Levels*. The review that have high probability to topic11 is mainly about easy levels that user feedback, and the review of topic8 is mainly about the daily level that the user plays. From the content we can conclude that topic11 and topic8 both are relevant to entity *Levels*, the same case with other entities. From all the topic probability matrix we find that, when the probability of a topic is below the 0.2, the topic is irrelevant to the entity, therefore we set our threshold as 0.2 in E-LDA, and get the result in TABLE VI.

For all the reviews that have assigned to different topics according to the entity, we can get the number of reviews in each topic of the entity. Then we implement our mixed sentiment computing algorithm, the result is shown in TABLE VII. We fetch 54,461 reviews from app *Bubble Shooter*, the E-LDA model outputs 21 topics of the reviews, we finally calculate that there are 26,141 reviews actually contain the meaningful information, and 7,842 reviews that are relevant to entity *Levels*, 5,228 reviews are about entity *Challenging*, and 3,398 reviews are about entity *Advertise*. We compute these entities based on these relevant reviews and its ratings, and our mixed sentiment computing method outputs the sentiment value which ranges from [-5,5], the mixed sentiment value for entity *Levels* is 2, the entity *Challenging* is 1, and the entity *Advertise* is -3.

TABLE VII
STRATIFIED STRUCTURE FOR BUBBLE SHOOTER REVIEWS

- Levels [sentiment: 2]: 7,842 reviews
  - topic 11:
    - …
  - topic 8:
    - …
  - …
- Challenging [sentiment: 1]: 5,228 reviews
  - topic15
    - …
  - topic6
    - …
  - …
- Advertise [sentiment: -3]: 3,398 reviews
  - topic10
    - …
  - Topic4
    - …
  - …

C. Evaluation of E-LDA

In general, there are two ways to evaluate topic model, the evaluation based on topic distribution and the evaluation based on Perplexity result, in this paper, we use Perplexity result [29] to evaluate E-LDA. Perplexity is a criterion in language models, it is used to evaluate the generalization ability of the model. In LDA it is computed in (3):

$$\mathrm{Per}plexity(\mathrm{D}_{test}) = \exp\left\{\frac{\sum_{d=1}^{M}\log p(\mathrm{w}_d)}{\sum_{d=1}^{M} N_d}\right\} \quad (3)$$

Where D_{test} denotes the testing set, which the set has M documents, and each document d has N_d words, and w_d denotes words vector. Perplexity value would decrease as the increase of $log\ p(w_d)$ value, which indicates that the smaller the perplexity value is, the better is the model. The Perplexity of E-LDA computing formula is described in (4).

[5] https://play.google.com/store/apps?hl=en

[6] http://stanfordnlp.github.io/CoreNLP/

$$\text{Per}plexity(\mathrm{D}_{test}) = \exp\left\{\frac{\sum_{d=1}^{M}\log p(\mathrm{w}_d \mid \mathrm{e})}{\sum_{d=1}^{M} N_d}\right\} \quad (4)$$

The number of topic K has a strong effect on the performance of E-LDA model, in order to get the proper value *K*, we test the Perplexity on the condition that *K*=10, 20, 30, 40, 50, 60, 70, 80, 100 respectively, the result is showed in Fig. 4. From the result we can conclude that Perplexity decease with the iteration accumulating in all numbers of topics, and eventually converge to a stable level. We can also conclude that in the same iteration, the Perplexity decreases as the topics increase, when *K* increase to 50 or larger, the Perplexity would grow at large. As the irrelative reviews would be assigned to one topic when *K* is too small, whereas the relative reviews would be divided into two different topics if *K* is too big. As a result, we choose to put the *K* =28 according to the data, when the Perplexity reaches the base point.

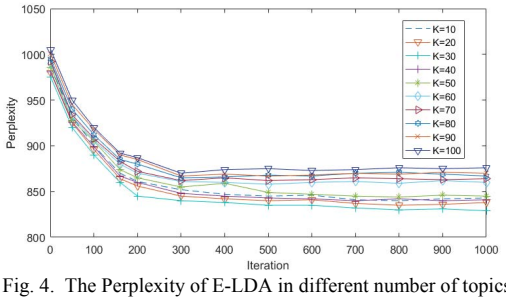

Fig. 4. The Perplexity of E-LDA in different number of topics

At the same time, we set iteration frequency as 1000, and put *K*=10, 20, 30, 40, 50, 60, 70, 80, 100 respectively, compare the Perplexity between LDA and E-LDA, Fig. 5. shows the result. For all different number of topics, the Perplexity of E-LDA is lower than that of LDA, which indicates that E-LDA has a better performance when we add an entity layer to LDA language model.

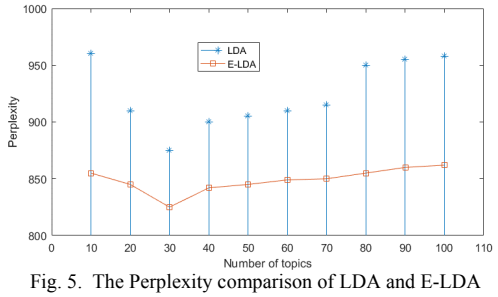

Fig. 5. The Perplexity comparison of LDA and E-LDA

In the end, we use recall, precision, and F-measure to evaluate our topics and entities that retrieved by SAR. The list below is the definition of reviews types that tested in SAR:

- True positives (TP): If it was automatically extracted from a review and was also manually identified in that review.
- False positives (FP): Reviews that were automatically associated to a topic in one of the topics and entities, but were not identified manually in that review.
- False negative (FN): Reviews that were manually identified in a topic but were not present in any of the extracted topics and entities associated to the review.
- True negative (TN): Reviews that were manually identified and also present in the extracted topics and entities.

Therefore, the recall *R* and precision *P* is computed in (5)

$$R = \frac{TP}{TP+FN}, \quad P = \frac{TP}{TP+FP} \quad (5)$$

F-measure of the system is defined as the weighted harmonic mean if its precision and recall, that is,

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \quad \alpha \in [0,1] \quad (6)$$

where α is the weight in (6), *F-measure* is high only when both recall and precision are high, it is equivalent to recall when α=0 and precision when α=1. The F-measure assumes values in the interval [0,1]. It is 0 when no relevant reviews have been retrieved, and is 1 if all retrieved reviews are relevant to same topics and entities. Table VIII summarizes the result. Compared to social app, two apps from game genre have lower recall and precision, we achieved the highest recall of 86% for Quora, this result probably due to higher quality of reviews that users give in social apps. For all the apps, we achieve the average recall 68%, and the average precision 64%, the result shows our approach to stratify the user reviews automatically is effective.

TABLE VIII
EVALUATION OF TOPICS AND ENTITIES

App	Recall	Precision	F-measure
Virtual Table Tennis	0.436	0.427	0.432
Bubble Shooter	0.534	0.528	0.531
Quora	0.864	0.832	0.847
Tumblr	0.823	0.714	0.765
YouTube	0.753	0.726	0.739
Speaker Boost	0.681	0.627	0.653
Average	0.682	0.642	0.661

VI. Related Work

SAR model that we proposed has three components: hot entity discovery from raw app reviews, a model grouping reviews into a stratified structure, and a mixed sentiment computing method for review entity. As a result, we focus the related work discussion in three areas: information extraction from app reviews, user reviews classification, as well as sentiment analysis on text.

A. Information Extraction of App Reviews

Manning et al. [13] introduced information retrieval technology in their book, and the foundations of statical natural language processing is also discussed on their book [7]. However, there are very few works in mining useful information from user's reviews with this knowledge. One of the earliest work is from Chandy et al. [20] who propose a simple latent model to identify spamming reviews on Apple AppStore. Recently, there are tools have been developed to analysis app reviews, helping developers discover most informative user reviews i.e. feature request, bug report, fraud reviews detection. Besides information retrieval, there are also other angles to analysis mobile app reviews [22, 24, 25, 32, 34].

Hu and Liu introduced an approach [2] to extract customers' opinion features from their reviews. A case study of user involvement in software evolution was conducted by Pagano and Brügge [28]. Carreno and Winbladh also introduce an approach for software requirements evolution, which is based on the analysis of user comments [21]. Claudia and Rachel [1]

proposed a prototype MARA (Mobile App Review Analyzer) to automatically retrieve request features of online reviews. The features extracted in their work based on linguistic rules which simply include some keywords that might be not sufficient for practical review analysis. Chen et al. proposed a computational framework AR-Miner [12] to extract and rank informative reviews at sentence level. Gao and Xu introduce a AR-Tracker which is capable of tracking the dynamics of mobile apps via user review mining [35]. Phong et al developed a keyword based tool MARK [37] for review analysis of mobile apps. The analyst can use MARK to list the reviews most relevant to a set of keywords, the tool can also draw trends over time of the selected keywords. As for fraud opinion detection among users' reviews, Akoglu et al. introduced a method that can exploits the network effect among reviewers and products [31]. Palomba et al. highlight the importance of user reviews in their work tacking crowdsourced reviews to support evolution of successful apps [33].

Based on the works above, our entity discovery algorithm of SAR is distinctive in three ways. Firstly, instead of simply extract feature request or bug reports, we get abstract hot entity that users care most, the entity is modified and abstract by the standard entity dictionary that we build from developer documents. Secondly, our hot entities are different from keywords, they focused on users' hot expectations on different aspects of the app. Finally, our method can be timing which is dynamically refresh from the dataset, always keep on the trend of users' thinking.

B. User Reviews Classification

There are ways to classify the text, one of earlier work that apply classification algorithms to mobile app reviews is the work of Antoniol et al. [23] who conducted experiments on classifying requests retrieved from reviews. They showed that alternating decision trees, naïve Bayes classifiers, and logistic regression can be used to accurately distinguish bugs from other kinds of issues. Herzig and Just discussed how misclassification impacts bug prediction [26]. Ohana and Tienery proposed a supervised learning method that can be applied to sentiment classification of user reviews [30]. Guzman et al. introduced a taxonomy for classifying app reviews into categories relevant for software evolution [36]. Maalej and Nabil introduced several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and ratings [38].

Our approach SAR for reviews classification distinguishes the previous work, we modify the topic model LDA [4], adding a hot entity layer which can stratify user reviews into a fine grain structure, which provides developers an easier and faster way to look into reviews, presenting reviews that relevant to entities fiercely discussed by users.

C. Sentiment Analysis on Reviews

Sentiment analysis on reviews can help developers know users' satisfaction about the app. Thelwall et al. proposed a method to detection the sentiment strength in short informal text [9]. Later they also analysis the sentiment strength on social web [10]. As for software evolution, Li and Zhang introduced an approach to analyze users' satisfaction toward software [5]. After that Fu and Lin conducted a sentiment analysis on user feedback in a mobile app store [14], to help develop understand why users hate the app. A study on free ios apps was conducted by Khalid and Shihab [17]. Before that Iacob et al. introduced a study of online reviews of mobile apps to figure out what users complain about [18]. Guzman and Maaleij proposed an automated approach that help developers filter, aggregate, and analyze user reviews, and then extract user sentiments about the identified features, giving them general score across all reviews [11]. Later, Guzman et al. presented a feature and sentiment centric retrieval approach which dynamically provides developers with a diverse sample of user reviews [39].

With respect to computation of the sentiment on user reviews, our novel model SAR combines the ratings and sentiment strength on short text, which is distinctive among the previous work, experiment shows that our mixed sentiment computing method in SAR presents the users' satisfaction of the app better than the single sentiment strength on reviews.

VII. CONCLUSION

The experiment indicates that SAR is effective in mobile app reviews analysis, proposing a nascent angle to get users feedback for mobile app developers. It can help developers improve their app design more efficient in three ways. Firstly, it can extract hot entities automatically from raw reviews, these entities are what users mostly concern about, it can let developers quickly understand what aspects of the app are being considered by users when they download the app. Secondly, SAR can stratify reviews into different topics according to the entity, it can provide the details of what the users are thinking about apps, making developers easier to find those informative content. Finally, SAR can assign sentiment value to each entity, which provide an intuitive way for developers to notice if the entity was favored by users. In addition, since SAR can dynamically absorb reviews, it can also be applied to monitor users' behaviors toward apps, making developer be the first person to know what aspects of app need to update or improve to attract latent users, this would make developers more competitive in their fellows. Another application of SAR is an analysis tool for users, when they start to find a new app, SAR can be applied to list the bad and good aspects of the app downloaded by previous users. There are also more works about SAR in future, the performance of E-LDA need to be improved in order to achieve a higher recall and precision.

VIII. ACKNOWLEDGMENT

We thank Claudia Iacob, Phong Minh and Jing Wang for their useful data source guide, as well as feedback and discussions. We also thank our experiment participants for their time and helpful comments. This work is supported by *National High-tech R&D Program (863 Program) (2015AA017202).*

REFERENCES

[1] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *MSR '13 Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 41-44. IEEE Press, May 2013.

[2] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining – KDD '04*, pages 755-760. AAAI Press, July 2004.

[3] M. Harman, Y. Jia, and Y.Zhang, "App store mining and analysis: MSR for app stores," in *Proc. of Working Conference on Mining Software Repositories* – MSR '12 pages 108-11, June 2012.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," in *The Journal of Machine Learning Research*, 3:993-1022, Mar. 2003.

[5] H. Li, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," in *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 2, pages 1093-1097. IEEE, 2010.

[6] W. Maalej and M. P. Robillard, "Patterns of knowledge in API Refernce Documentation," in *IEEE Transactions on software Engineering*, 39(9):1264-1282, 2013.

[7] H. Manning, Christopher D., Schütze, "*Foundations of statistical natural language processing*," in MIT Press, 1991.

[8] D. Pagano and W. Maaalej, "User feedback in the appstore: an empirical study," in *Proc. of the International Conference on Requirements Engineering – RE '13*, pages 125-134, 2013.

[9] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," in *Journal of the American Society for Information Science and Technology*, 61(12):2544-2558, Dec. 2010.

[10] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," in *Journal of the American Society for Information Science and Technology*, 63(1): 163-173. Jan. 2012.
E. Guzman, W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Requirements Engineering Conference (RE)*, 20144 IEEE 22nd International. Aug. 2014. [Online]. Available:http://ieeexplore.ieee.org/document/6912257/

[11] N. Chen, J. Lin, S. C. H. 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*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767-778. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2568225.2568263

[12] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," in Cambridge university press Cambridge, 2008, vol. 1. [Online] Available: http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[13] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and data Mining, ser. KDD '13*. New York, NY, USA: ACM, 2013, pp. 1276-1284. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488202

[14] R. Vasa, L. Hoon, K. Mouzakis, and A. Nohuchi. A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference, ser. OzCHI '12*. New York, NY, USA: ACM, 2012, pp. 241-244. [Online]. Available: http://doi.acm.org/10.1145/2414536.2414577

[15] L. Hoon, R. Vasa, J.-G Schneider, and J. Grundy, "An analysis of the mobile app review landscape: Trends and implications," in Tech. rep., *Faculty of Information and Communication Technologies, Swinburne University of Technology*, Melbourne, Australia, Tech. Rep., 2013

[16] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile users complain about? a study on free ios apps," 2014.

[17] C. Iacob, V. Veerappa, and R. Harrison, "what are you complaining about? a study of online reviews of mobile applications," in *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 2013, p. 29.

[18] G. Bavota, M. Linares-Vasquez, C. Bernal-Carden, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "The impact of api change- and fault-pronesness on user ratings of Android apps," *Software engineering*, IEEE Transactions on, vol. 41, no. 4, pp. 384-407, April 2015.

[19] R. Chandy and H. Gu, "Identify spam in the ios app store," in *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, ser. WebQuality, '12. New York, NY, USA: ACM, 2012, pp. 56-59. [Online]. Available: http://doi.acm.org/10.1145/2184305.2184317

[20] L. Galvis Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Software Engineering (ICSE), 2013 35th International Conference* on, May 2013, pp. 582–591.

[21] M. Gomez, R. Rouvoy, M. Monperrus, and L. Seinturier, "A recommender system of buggy app checkers for app store moderators," Ph.D. dissertation, Inria Lille, 2014.

[22] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: A text-based approach to classify change requests," in *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds,* CASCON '08, pages 23:304–23:318. ACM, 2008.

[23] M. Bano and D. Zowghi, "A systematic review on the relationship between user involvement and system success." Information & Software Technology, 58:148–169, 2015.

[24] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics." Reseach Note RN/14/10, UCL Department of Computer Science, 2014.

[25] K. Herzig, S. Just, and A. Zeller, "It's Not a Bug, It's a Feature: How Misclassi• cation Impacts Bug Prediction," in *Proceedings of the 2013 International Conference on Software Engineering*, pages 392–401. IEEE Press, 2013.

[26] T. Johann and W. Maalej. "Democratic mass participation of users in requirements engineering?" In *Requirements Engineering Conference (RE)*, 2015 IEEE 23rd International, 2015.

[27] D. Pagano and B. Brügge, "User involvement in software evolution practice: A case study," in *Proceedings of the 2013 International Conference on Software Engineering*, pages 953–962. IEEE Press, 2013.

[28] F. Li, H. Shen, T. He, "Tag-Topic model for semantic knowledge," in Natural Language Processing and Knowledge Engineering (NLP-KE), 2011 7th International Conference on, 27-29 Nov. 2011.

[29] B. Ohana, B. Tierney, "Supervised learning methods for sentiment classification with RapidMiner," in *Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference,* on 26-28 Aug. 2010.

[30] L. Akoglu, R. Chandy, C. Faloutsos, "Opinion fraud detection in online reviews by network effects," in *Proceedings of ICWSM*, 2013

[31] X. Xu, K. Dutta, A. Datta, "Functionality-based mobile app recommendation by identifying aspects from user reivews," in *Thirty Fifth International Conference on Information Systems,* Auckland 2014

[32] F. Palomba, M. Linares, G. Bavata, R. Oliveto, "User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference* on 29 Sept.-1 Oct. 2015.

[33] M. Liu, C. Wu, X. Zhao, C. Lin, X. Wang, "App relationship calculation: an iterative process," in *IEEE Transactions on Knowledge & Data Engineering*, vol.27, no. 8, pp. 2049-2063, Aug. 2015.

[34] C. Gao, H. Xu, J. Hu, Y. Zhou, "AR-Tacker: tack the dynamics of mobile apps via user review mining," in arXiv:1505.04657 [cs.IR]. [Online]. Available: http://arxiv.org/abs/1505.04657

[35] E. Guzman, M. EI-Halaby, B. Bruegge, "Ensemble methods for app review classification: an approach for software evolution," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference*, on 9-13 Nov. 2015.

[36] P. Vu, T. Nguyen, H. Pham, T. Nguyen, "Mining user opinions in mobile app reivews: a keyword-based approach," in arXiv:1505.04657 [cs.IR].

[37] W. Maalej, H. Nabil, "Bug report, feature request, or simply raise? On automatically classifying app reviews," in *IEEE International Requirements Engineering Conference*, pp 116-125, 2015.

[38] E. Guzman, O. Valeo, B. Bruegge, "Retrieving diverse opinions from app reivews," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, May 18-19, 2013, San Francisco, CA, USA.

[39] Z. Xie, S. Zhu, "AppWatcher: unveiling the underground market of trading mobile app reviews," in *8th ACM Conference on Security and Privacy in Wireless and Mobile Networks,* NYC, USA, June 24-26, 2015.

[40] *T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Ef• cient estimation of word representations in vector space," CoRR, vol. abs/1301.3781, 2013.* [Online]. Available: http://arxiv.org/abs/1301.3781