Tong YX, She J, Chen L. Towards better understanding of app functions. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(5): 1130–1140 Sept. 2015. DOI 10.1007/s11390-015-1588-0

Towards Better Understanding of App Functions

Yong-Xin Tong [1,2,*] (童咏昕), *Member, CCF, ACM, IEEE*, Jieying She [3] (佘洁莹), *Student Member, IEEE*, and Lei Chen [3] (陈 雷), *Member, ACM, IEEE*

[1] *State Key Laboratory of Software Development Environment, School of Computer Science and Engineering Beihang University, Beijing 100191, China*

[2] *International Research Institute for Multidisciplinary Science, Beihang University, Beijing 100191, China*

[3] *Department of Computer Science and Engineering, The Hong Kong University of Science and Technology Hong Kong, China*

E-mail: yxtong@buaa.edu.cn; {jshe, leichen}@cse.ust.hk

Received November 16, 2014; revised July 22, 2015.

Abstract Apps are attracting more and more attention from both mobile and web platforms. Due to the self-organized nature of the current app marketplaces, the descriptions of apps are not formally written and contain a lot of noisy words and sentences. Thus, for most of the apps, the functions of them are not well documented and thus cannot be captured by app search engines easily. In this paper, we study the problem of inferring the real functions of an app by identifying the most informative words in its description. In order to utilize and integrate the diverse information of the app corpus in a proper way, we propose a probabilistic topic model to discover the latent data structure of the app corpus. The outputs of the topic model are further used to identify the function of an app and its most informative words. We verify the effectiveness of the proposed methods through extensive experiments on two real app datasets crawled from Google Play and Windows Phone Store, respectively.

Keywords app function, document, topic model

1 Introduction

In recent years, with the rapid development of social medias[1-3] and Online To Offline (O2O) marketing model[4-6], apps development for mobile devices and web platforms has quickly become a million-dollar industry. IT giants such as Apple and Google have opened their own online app markets, i.e., Apple App Store and Google Play, in order to build an ever-growing app ecosystem. As of June 2014, there have been 1 200 000 apps on Apple App Store while there have been about 400 000 apps available online in Android markets. With the participation of enthusiastic application developers who are inspired by the diverse real-life needs from users, the app market has changed into online business mode revolutionarily, which demonstrates its unique characteristics compared with traditional online markets that sell books, albums or movies. For example, in order to attract customers and encourage users to download the apps, app developers usually describe their products with eye-catching names and app descriptions, which result in names that are not descriptive enough to reveal the apps' real functions. With a large number of apps, the current app market, which is still growing rapidly, still only relies on coarse-granulated categories to describe an app's general functions, which may lead to misunderstanding and confusion. For example, we present some statistics of app categories in Appendix, from which we can observe that the app categories, Brain & Puzzle, Arcade & Action, Communication, and Entertainment, are the most

Regular Paper

Special Section on Social Media Processing

This work is supported in part by the Hong Kong RGC Project under Grant No. N_HKUST637/13, the National Basic Research 973 Program of China under Grant No. 2014CB340303, the National Natural Science Foundation of China under Grant Nos. 61328202 and 61502021, Microsoft Research Asia Gift Grant, Google Faculty Award 2013, and Microsoft Research Asia Fellowship 2012.

*Corresponding Author

popular ones and the category distribution is quite biased. Hence, with the noisy app descriptions and the coarse-granulated app categories, it is usually very hard for users to identify the real functions of the apps.

Therefore, it is necessary to develop an effective method to clearly reveal the real functions of the apps to users. However, few studies have been done to address this problem.

Through an extensive analysis of an app corpus crawled from Google Play, we observe that the app descriptions have some unique features. First, app descriptions usually have informal words and sentences, which are not useful in revealing the functionality of an app. Second, each app description contains technical terms from its own niche, which demonstrate burstiness. The burstiness is the phenomenon that if a word appears in the description, it is likely to appear again. Finally, though each of the apps belongs to a specific category, they are not isolated from each other. In an app's download page, the app store also recommends other kinds of apps based on three kinds of relations: also-installed, also-viewed, and same-developer. In this paper, we utilize such features to address the aforementioned challenges via the paradigm of Bayesian network. Specifically, we propose the App Generative model (AGM). AGM utilizes the Pitman-Yor (PY) process to create the power-law distribution of terms, which essentially models the burstiness of words in the app description. The contributions of this paper are summarized as follows.

1) We propose and study the new problem of app functionality inference, which is critical for improving the performance of app search engines.

2) We propose a novel generative model to capture the unique features of app corpus.

3) We conduct extensive experiments on a large-scale app corpus and the experimental results verify the effectiveness and the superiority of the proposed model.

The rest of the paper is organized as follows. In Section 2, we review the related work. In Section 3, we analyze the crawled app information. In Section 4 and Section 5, we discuss the strategy of app functionality inference and discovery, respectively. In Section 6, we present the experimental results. Finally, the paper is concluded in Section 7.

2 Related Work

Bayesian network is becoming more and more popular in text analyzing. Blei *et al.*[7] proposed the pioneering Latent Dirichlet Allocation (LDA) to analyze electronic archives. Griffiths and Steyvers[8] reported that LDA is effective to find scientific topics. Following LDA, many topic models that specialize in different tasks are further proposed. Jo and Oh[9] proposed two generative models to analyze the aspects and sentiments of online reviews. Sato *et al.*[10] proposed a topic model based on the Pitman-Yor (PY) process, which can capture the property that word distribution follows power-law in a document. Wang *et al.*[11] proposed a location aware topic model to explicitly capture the relationship between locations and words. Moreover, Yin *et al.*[12] studied the problem of discovering and comparing geographic topics from GPS-associated tweets. They proposed and compared three ways of modeling geographic topics: location-driven model, text-driven model, and a joint model that combined location and text. Jiang *et al.*[13] proposed two models to analyze search engine query log from the perspective of geographic topics. Moreover, Jiang *et al.*[14] proposed three models to discover the latent structure of the query log data without using the classical Click Graph-Based approaches. Sizov[15] proposed Bayesian models for characterization of social media by combining text features with spatial knowledge. Eisenstein *et al.*[16] proposed a multi-level generative model that reasons jointly about latent topics and geographic regions. Jiang *et al.*[17-18] proposed a new query suggestion paradigm, Personalized Query Suggestion with Diversity Awareness (PQS-DA) to effectively combine diversification and personalization into one unified framework. Hao *et al.*[19] proposed a location-topic model to mine location-representative knowledge from a large collection of travelogues. Our proposed App Generative model is specialized for the scenario of app analysis.

Though the aforementioned probabilistic topic models are successful in analyzing various types of data, very little work has been done to analyze the app data from the perspective of topic modeling. To the best of our knowledge, our work is the first one that focuses on inferring the real functions of apps. Our work can pave the way for many downstream applications such as app search and app recommendation.

3 App Information

In this section, we analyze the app information that is publicly accessible on Google Play. The analysis is later utilized to develop our App Generative model. An example of the crawled app information is presented in Table 1. As shown in Table 1, each app contains attribute information, such as category, developer, average rating, price, and the installed number, and link information, which includes three kinds of links: same-developer, also-viewed, and also-installed. We first investigate the textual similarity between an app and its neighbors, which is presented in Table 2. The result reveals that apps having the same-developer relation are the most similar to each other in both terms of title and description. Apps that have also-installed relation are the most distant in both terms of title and description. The statistics indicate the following. 1) The same developer tends to develop apps that have similar titles and descriptions. It is consistent with our intuition that the same developer usually releases a series of apps, which usually have similar titles. The developer may also have the same description styles for all her/his apps, making the description similar to each other. 2) The also-installed relation may link up apps that have different functions and that are developed by different developers, which contribute to the large distance of title and description in such apps.

Table 1. Example of an App Information in Google Play

App name	Clash of Clans
App description	Introducing: Clan Wars! Crush enemy clans in clan versus clan battles. Clash of Clans is an epic combat strategy game. Build your village, train your troops and battle with millions of other players online!...
Category	Strategy
Developer	Supercell
Average rating	4.6
Rating users	6 260 285
Price	Free
Installs	50 000 000∼100 000 000
Same-developer	Hay Day, Boom Beach, ...
Also-viewed	Clash of Lords 2, Castle Clash, Battle of Zombies, Calculator for Clash, ...
Also-installed	Legend of Empire, Armies of Dragons, Boom Beach, Clash of Lords 2, ...

Table 2. Textual Similarity

Relation	Ave. Edit Dis. (Title)	Ave. Cosine Sim. (Text)
Same-developer	16.557 869 75	0.350 160 113
Also-viewed	17.685 921 19	0.335 543 795
Also-installed	17.699 700 17	0.271 899 755

We next observe that on Google Play, there are about 30 categories, which provide a coarse-granular classification system of the apps' functions. Another phenomenon observed is the burstiness of words used in the app description. We split the words into three categories: common, average, and rare, based on how often they appear in the query log. The common words are the 500 most frequent words; they represent 0.056 8% of the words in the vocabulary and 30.57% of the appearance. The average words are the next 5 000 most common words; they represent 0.567 0% of the vocabulary and 28.46% of the appearance. The rare words are the rest of the vocabulary and account for 40.95% of the appearances. In Fig.1, we present the distributions of the three categories of words in our dataset. Interestingly, the curves of the common words and the average words are parallelly close to each other and have similar decay rates. However, the decay rate of the rare words is much lower than those of the common words and average words. Once a rare word appears, the probability that it will appear multiple times is much higher than that of the common words and the average words.

Table 3. Notations

Notation	Description
D	Number of documents
N	Average number of words
K	Number of topics
z	A topic
z_i	Topic of word i
$\boldsymbol{z}_{-i}$	Topic assignments for all words except word i
w	A word
$\boldsymbol{w}$	Word list representation of the corpus
θ	Multinomial distribution over topics
ϕ	Multinomial distribution over words
$N_{j,k,v}$	Number that the word v is generated by the topic t in document j
B	Number of bullets
P	Number of patterns

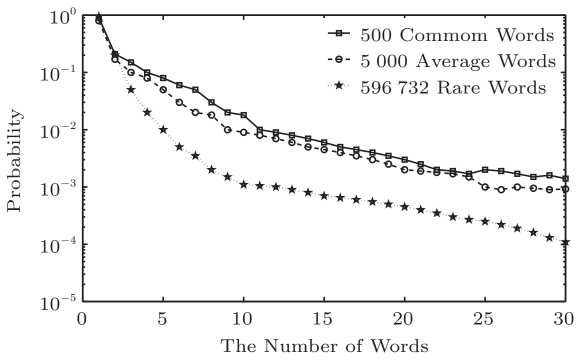


Fig.1. Power-law phenomenon of the words in app description.

In summary, we investigate different characteristics of the app corpus, which will be utilized in our App Generative model.

4 App Function Inference

In this section, we first present the App Generative model in Subsection 4.1 and then show how to infer the parameters of the models in Subsection 4.2.

4.1 App Generative Model

The App Generative model (AGM) captures three properties of an app: the power-law word distribution, the link information, and the presence of multiple topics. This model uses the Pitman-Yor (PY) process[20] to model the power-law word distribution, which is one of the most adaptive processes for document modeling due to its exchangeability property. The PY process is a stochastic process generalized from the Dirichlet process[21]. The PY process has a concentration parameter γ and a discount parameter d that control the power-law property. The discount parameter places priority on new words that induce long tails in the power-law distributions which are effective for modeling distributions that have many words with frequency of 1. The PY process has a stochastic metaphor called the Chinese restaurant process (CRP), which is a process for establishing customers' seating arrangement in a restaurant, where the number of customers seated at tables follows a power-law distribution.

Besides the power-law word distribution, the link information of apps and the presence of multiple topics can be naturally incorporated into the generative assumption of AGM. The generative process of AGM is shown in Algorithm 1. The logic is as follows. Each word is related to a specific topic that covers the app description. The instance of a word is generated from a topic-word distribution. We also assume that we use words that have already appeared more often, i.e., when we select a word, we consider not only the context-specific topics but also the words that we previously used. This is modeled by using the table in CRP and the topic-word distributions as follows. One who tends to use words that have already appeared is modeled as a customer who sits at an existing table more often. One who has a large vocabulary and tends to select words from a topic-word distribution is modeled as a customer who prefers to sitting at a new table. Our model generates a topic at each table in a CRP representation of the document. While the number of generated topics is equal to the number of words in LDA, the number of topics in our model is equal to that of tables rather than words. Therefore, we introduce latent variable $z_{j,k}$ to denote the topic that is assigned to the k-th table in document j. In our model, the seat arrangement of a customer is drawn according to the following formula: the k-th occupied table has the probability of $\frac{N_{j,v}-dK_{j,v}}{\gamma+N_j}$ to be arranged and an unoccupied table has probability $\frac{\gamma+dK_{j,\cdot}}{\gamma+N_j}$ to be arranged. If a customer sits at an unoccupied table, the customer will then sample a topic from the topic distribution of the document, and also a word from the word distribution of the sampled topic and the category information from the sampled topic.

Algorithm 1. Generative Process of App Generative Model

```
1  for each topic $t \in \{1, ..., T\}$ do
2      Draw $\phi_t \propto Dirichlet(\beta)$;
3  for each document $d \in \{1, ..., D\}$ do
4      Draw a linked app $d'$;
5      Draw current app's topic distribution
       $\theta_d \propto Dirichlet(\alpha)$;
6      for each word $w \in \{1, ..., N_d\}$ do
7          Sit at the $k$-th occupied table in
           proportion to $N_{j,k} - d$;
8          Sit at an unoccupied table in proportion
           to $\gamma + dK_j$;
9          Draw topic $z \propto Multinomial(\theta_d + \theta_{d'})$;
10         Draw word $v \propto Multinomial(\phi_z)$;
```

4.2 Parameter Inference

Inspired by [10], the predictive probability of observing a new word, given the words, topics and the seating arrangements in documents, is:

$$\begin{aligned} & P(w_j^{\text{new}} = v|w, z, x) \\ = & \frac{N_{j,v} - dK_{j,v}}{\gamma + N_j} + \frac{\gamma + dK_{j,\cdot}}{\gamma + N_j} \times \\ & \sum_{t=1}^{T} \sum_{d'} \left(\frac{N_{j,t} + \alpha_t}{N_j + \alpha_0} + \theta_{d'}\right) \frac{N_{t,v} + \beta}{N_{t,\cdot} + V\beta}, \end{aligned}$$

where z is the set of topics, x is the set of new topics generated by CRP, d is a document, d' stands for the linked document of the current document, $N_{j,v}$ denotes the number of customers serving word v, which indicates the frequency of word v in document j, $N_{j,t}$ is the number of the topic t appearing in document j,

N_j is the number of words in document j, $N_{t,v}$ denotes the number of word v generated by topic t, $N_{t,\cdot}$ is the number of all words that are assigned to topic t, $K_{j,v}$ is the number of tables serving word v in document j, $K_{j,\cdot}$ is the number of all tables in document j, α_t, β and γ are prior parameters, $\alpha_0 = \sum_{t=1}^{T}\alpha_t$, $\theta_{d'}$ is the K-dimensional probability distribution of topics in the new document d', and V is the number of words.

The probability of generating a topic at a new table is given by:

$$P(z_{j,k^{\text{new}}} = t|w_{j,i} = v, x_{j,i} = k^{\text{new}}, z, w^{-j,i}, x^{-j,i})$$
$$= \sum_{d'}\left(\frac{N_{j,t}+\alpha_k}{K_j+\alpha_0} + \theta_{d'}\right)\frac{N_{t,v}+\beta}{N_{t,\cdot}+V\beta},$$

where $w^{-j,i}$ indicates the set of all words except the word i in document j, and $x^{-j,i}$ indicates the set of all new topic except topic i in document j, $z_{j,k^{\text{new}}}$ denotes the topic that is assigned to the k-th new table in document j, $w_{j,i}$ is the i-th word in document j, $x_{j,i}$ is topic i in document j, and K_j is the number of all topics in document j.

According to aforementioned formulas, the inference algorithm regarding the App Generative model is shown in Algorithm 2.

Algorithm 2. Inference Algorithm of AGM

```
1 repeat
2     for each document $j \in \{1, ..., D\}$ do
3         for each word $i \in \{1, ..., N_j\}$ do
4             With a probability proportional to $N_{j,k,v}$, remove a word from the $k$-th table in document $j$;
5             if the $k$-th table becomes unoccupied then
6                 Remove the table from document $j$, decrement $N_{z_{j,k},v}$, where $v$ is the word at the table;
7             Draw a new topic, with a probability proportional to $\max(0, N_{j,k,v} - d_t)$, sit word at the $k$-th table serving word $v$ in document $j$;
8             With a probability proportional to $(\alpha + dK_{j,v}\frac{N_{t,v}+\beta}{N_{t,\cdot}+V\beta})$, sit word at a new table, draw topic $z_{j,k^{\text{new}}}$ for the new table by $P(z_{j,k^{\text{new}}} = t|w_{j,i} = v, x_{j,i} = k^{\text{new}}, z, w^{-j,i}, x^{-j,i}) = \sum_{d'}(\frac{N_{j,t}+\alpha_k}{K_j+\alpha_0} + \theta_{d'})\frac{N_{t,v}+\beta}{N_{t,\cdot}+V\beta}$, increment $N_{jt}$, $N_{tv}$;
9 until enough iterations;
```

Time Complexity Analysis. According to Algorithm 2, we can know that the time complexity is $O(IDNK)$, where I is the number of iterations of the sampling process. In each iteration, D is the number of documents, N is the average number of words in the documents, and K is the upper bound of the number of the learned topics.

5 App Function Discovery

In this section, we introduce two types of app function discovery methods using the aforementioned AGM model. In Subsection 5.1, we show how the function of apps can be represented by informative words through the discovered latent topics of AGM. We then show how the features of app developers can be captured according to the discovered latent topics via the AGM model in Subsection 5.2.

5.1 Function Word Extraction

After applying the model to the app data, we obtain each app's topic distribution θ and each topic's multinomial distribution over words ϕ. The score of a word w is calculated as follows:

$$Score(w) = \sum_{k=1}^{K} P(w|\phi_k) \times \theta_k.$$

After obtaining the scores of the words, we rank them in non-ascending order and use the top k of them as the words that are the most informative ones for the function of the app. Then, these informative words are used to represent the function of the app. In particular, more details of the experiment on function word extraction over real apps on Google Play are discussed in Subsection 6.1.

5.2 App Feature Extraction

The free-text descriptions of each app are essentially non-structured data. However, we observe that app developers tend to put some features in the description. In this subsection, we provide an app description template to capture the features in the app descriptions. We observe that these technically detailed features usually start with FEATURES or Features, after which, a sentence usually starting with "$\star$", "*", "**", "-", "###" describes an aspect of the features. Therefore, we use a series of regular expressions to extract the features in app descriptions. After the feature extraction, the description is further classified into general description and specific features. Specifically, we utilize the following feature indicators: $\{-, +, *, 1.(2.,...), \surd, \bullet, >>\}$. The pseudo-code of feature extraction is shown in Algorithm 3.

Algorithm 3. Feature Extraction of App Description

```
1  if find regular pattern feature?:? with Case
   Insensitive in description then
2      pattern_index ← index_of_the_pattern;
3      Find the first bullet occurring in the bullets
       set in the following text after pattern_index;
4      bullet_index←index_of_first_bullet;
5      if the distance between bullet_index and
       pattern_index is not too large (less than 20
       words) then
6          feature_list=[];
7          repeat
8              Find the next bullet;
9              feature←text between the two bullets;
10             feature_list←feature_list∪feature;
11         until end of the feature list;
12     else
13         feature_list=[];
14 else
15     feature_list=[];
```

Time Complexity Analysis. According to Algorithm 3, we can know that the time complexity is $O(PB)$, where P is the number of patterns, and B is the number of bullets.

6 Experiments

In this section, we present the experimental results. The experimental data are crawled from two real apps markets, Google Play in January 2012 and Windows Phone Store in April 2013, respectively. In Subsection 6.1, we present some topics discovered by AGM from the app corpus. In Subsection 6.2, we quantitatively compare AGM with some existing topic models in the two datasets. We also evaluate the performance that AGM is extended to search engines in Subsection 6.3. In Subsection 6.4, we evaluate the effectiveness of the proposed algorithm in extracting functionality-related words.

6.1 App Topics and Function Words

An informal but important measure of the success of probabilistic topic models is the plausibility of the discovered search topics. For simplicity, we use the fixed symmetric Dirichlet distribution like [2], which demonstrates good performance in our experiments. Hyper-parameter setting is well studied in probabilistic topic modeling and is beyond the scope of this paper. Interested readers are invited to refer to [22] to find a more detailed discussion. The top-5 keywords of the four topics discovered by AGM are presented in Table 4. We can see that the discovered topics consist of semantically related words, which include different aspects of the apps and show the effectiveness of AGM. For example, topic 1 consists of words about military and war.

We also show some top-ranked function words discovered by AGM in Table 5. As discussed in Subsection 5.1, these informative words can be extracted to represent functions of apps. For example, in the description of Angry Birds, the first three words, bird, angry and pig, describe the leading roles and the main story of this game. The five words in the first column summarize the main features of this app. Words in the other four columns also show the main functions of their corresponding apps.

6.2 Quantitative Comparison

In this subsection, we present the quantitative evaluation of AGM. Specifically, the evaluation is conducted based on the concept of perplexity measurement[23]. Before we elaborate the details of the

Table 4. Examples of Topics Discovered by AGM

Topic 1		Topic 2		Topic 3		Topic 4	
Keyword	Probability	Keyword	Probability	Keyword	Probability	Keyword	Probability
Tank	0.047 892	Bird	0.027 354	Finance	0.010 348 0	Food	0.094 581
Gun	0.042 945	Animal	0.020 183	Bank	0.009 170 0	Restaurant	0.052 139
Player	0.034 387	Piggy	0.017 532	Account	0.008 235 0	Chicken	0.049 223
Charge	0.029 219	House	0.013 725	Cash	0.006 251 2	Hamburger	0.021 740
Enemy	0.028 003	Block	0.010 281	Transfer	0.005 216 0	Beef	0.020 752

experiment, we first present its definition as below:

$$\begin{aligned} & Perplexity1(\theta, \phi) \\ = & \prod_{d=1}^{D} \prod_{i=1}^{N_d} p(w_i|\theta, \phi)^{\frac{-1}{\Sigma_{d=1}^{D}(N_d)}}, \quad (1) \\ & Perplexity2(\theta, \phi) \\ = & \prod_{d=1}^{D} \prod_{i=P+1}^{N_d} p(w_i|\theta, \phi, w_{1:P})^{\frac{-1}{\Sigma_{d=1}^{D}(N_d - P)}}, \quad (2) \end{aligned}$$

where $w_{1:P}$ are observed words.

Table 5. Examples of Function Words

Angry Bird	Angry Piggy	Candy Crush	Clash of Clan	Plant V.S. Zombie
Bird	Pig	Candy	Clan	Plant
Angry	Bird	Sweet	Battle	Zombie
Pig	Send	Crush	Clash	Fight
Egg	Hard	Saga	Fight	Bean
Eagle	Stone	Delicious	Army	Sun

The difference between *Perplexity*1 and *Perplexity*2 is that the former one is used to describe the *held-out* perplexity of the learned model ϕ, and the latter one is used to evaluate the effectiveness of prediction of AGM. Especially, better generalization performance is indicated by a lower perplexity.

Through an extensive survey, we cannot find any probabilistic model that is designed for analyzing apps, and thus we carefully choose several state-of-the-art models that are general enough to be applied to app analysis. Specifically, we use LDA[7] and PLink-LDA[24] as baselines. We compare AGM with the baselines by a 10-fold cross validation on two datasets crawled from Google Play and Windows Phone Store respectively, each of which includes 10 000 apps, and use (1) and (2) to calculate *perplexity*1 and *perplexity*2 of each model, respectively.

Fig.2(a) and Fig.2(d) show the average *perplexity*1 of each model on both datasets. We can observe that AGM achieves significantly lower perplexity, which indicates that AGM provides a better fit for the app data than the other models on both datasets.

We also measure how effective the proposed model is in terms of predicting the future app functions based on a portion of the available app functions. Specifically, given some words from a log of app description documents, we try to find out which model predicts the term distribution of the remaining apps more accurately. In particular, 80% of the data are used as training data and the remaining 20% are used as test data. We calculate the perplexity of each model according to (2) and summarize the comparison results in Fig.2(b) and Fig.2(e). As shown in the two subfigures, AGM always shows good capability in predicting the functions of the apps that will be released in the future.

Similar to [12], we also use KL-divergence to measure the differences of the discovered topics.

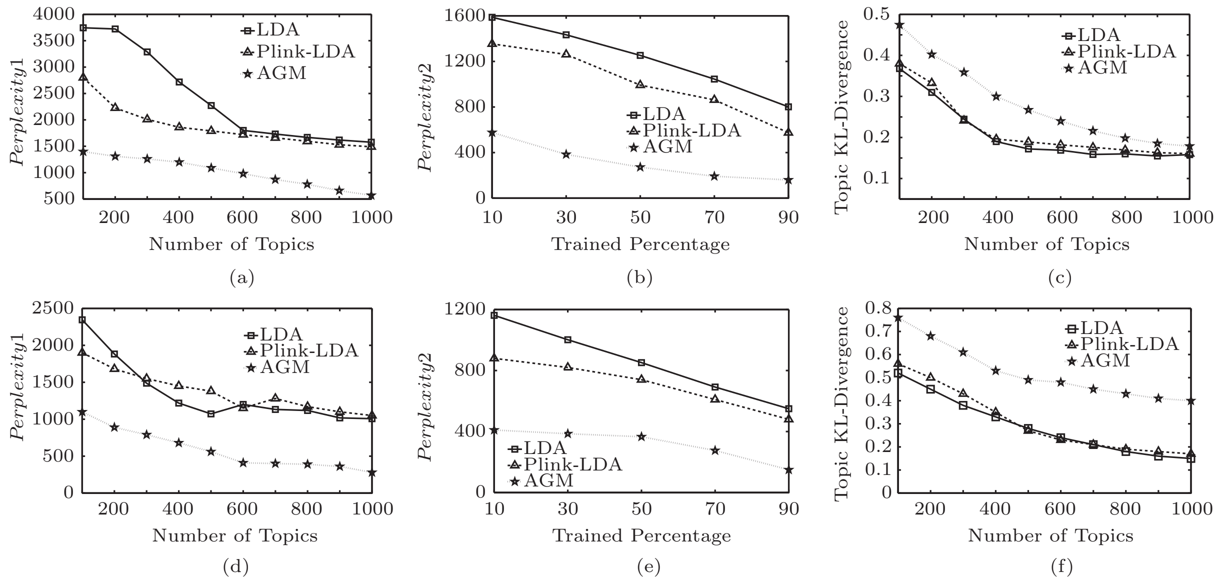


Fig.2. Quantitative comparison of *Perplexity*1, *Perplexity*2 and KL-divergence. (a) *Perplexity*1 in Google Play. (b) *Perplexity*2 in Google Play. (c) KL-divergence in Google Play. (d) *Perplexity*1 in Windows Phone. (e) *Perplexity*2 in Windows Phone. (f) KL-divergence in Windows Phone.

Fig.2(c) and Fig.2(f) show the average distance of KL-divergence regarding the term distributions of all pair-wise discovered topics in both datasets. We observe that AGM always has the highest KL-divergence on the two datasets, and the two baselines have similar KL-divergences. Since the lower average KL-divergence indicates the smaller difference between the discovered topics, the aforementioned results confirm that the word distributions and the topic distributions generated by AGM are more distinctive than those generated by the two baselines.

Finally, we evaluate whether our algorithm can successfully extract appropriate words for describing the function of an app. First of all, we employ 20 volunteers to label the functions of the apps in our used datasets as the ground truth in our experiments. The baseline is the TF-IDF method, which extracts the top-k words having the highest TF-IDF values. The experimental results on both the two datasets are presented in Fig.3(a) and Fig.3(b). As shown in the results, our proposed AGM method obtains better precision results than the traditional TF-IDF method, showing that our method is superior in detecting useful keywords for describing the functions of apps.

6.3 Extension to Search Engines

In this subsection, we evaluate the performance of AGM when it is extended to search engines. We first build a search engine prototype based on Lucene, an open-source software package for information retrieval. The prototype integrates the proposed AGM in this paper and the retrieval scheme of Lucene. Then, we evaluate whether the proposed AGM can enhance the quality of search engine through comparing the ranking results of our prototype search engine with those of the following two commercial app search engines.

• Google Play: the app ranking result of Google Play.

• Windows Phone Store: the app ranking result of Windows Phone Store.

The evaluation is conducted by a user-based method. We employ 20 volunteers, each of who is invited to conduct 30 searches on our prototype search engine and the above two commercial app search engines. Moreover, each volunteer is required to judge the top 50 ranking results for each query in each search engine by marking a satisfaction score, which includes three levels of relevancy (Good, Fair, and Poor) and is used to measure the relevance of the results to the semantic needs of users for each search engine. The Good level with score value of "1" indicates that the ranking result is positively relevant to users' needs, and the Poor level with score value of "-1" indicates that the ranking result is irrelevant to users' needs. The Fair level with score value of 0 is considered as that the volunteer is neutral regarding the ranking result. Finally, for each search engine, we calculate the average precisions by normalizing the summed satisfaction scores for different top-k queries. Fig.3(c) reports the average precisions of different top-k queries. We can observe that our prototype search engine integrating AGM provides better ranking quality under certain circumstances than the two commercial search engines of Google Play and Windows Phone Store. In particular, our prototype search engine performs better when the parameter k of the top-k queries is larger than 30. The results confirm again that AGM can provide a better fit for apps data.

6.4 Function Inference Comparison

In this subsection, we verify the effectiveness of our app feature extraction method. As discussed in Subsection 5.2, our proposed approach (Algorithm 3) can automatically extract the features from the app description. Firstly, we show a real example of app feature extraction in Table 6. According to crawled app description of "Angry Piggy" in Google Play, Table 6 presents

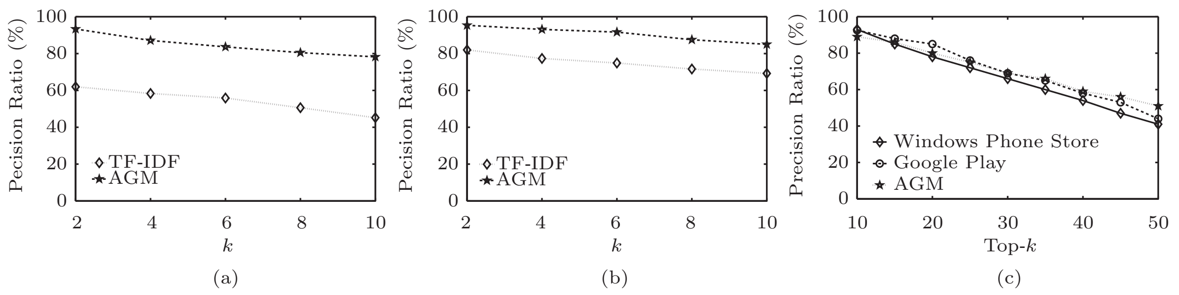


Fig.3. Evaluations of app function words and extension to search engines. (a) App function in Google Play. (b) App function in Windows Phone. (c) Quality of search engines.

nine features of this app extracted by our proposed algorithm.

Table 6. Example of the App Feature Extraction for Angry Piggy

Number	Feature
1	Transferred from a hot iOS game with the same title
2	Alternative controls: either gravity sensitivity or virtual buttons that are easy to use and just feel right
3	Over 40 challenging and puzzling physics-based levels
4	Over 40 Achievements and Leaderboards to compete with your friends
5	Switch between characters to perform best teamwork
6	Explore the unique islands from Ukurawe to Hoddanfield: unlockable challenges, secrets and rewards
7	Dramatic and dreamy art design which looks amazing on your device
8	Original soundtrack with an emotive piano tune
9	And more fun!

Furthermore, we recruit four human judges to select the 10 most informative terms from the app description. The human-prepared data is further used as ground truth. Then we utilize the Jaccard similarity to evaluate the distances between the ground truth and that generated by different baseline methods. In terms of the scale of the experiments, we select 10 000 apps as the experimental data. Experimental results show that the average precision of our proposed algorithm is greater than or equal to 93.8%.

6.5 Summary of Experiments

Summary. In this subsection, we summarize our experimental findings.

• AGM can discover semantic information from different aspects of the apps, and informative words regarding the apps functions can be well extracted by our proposed solution.

• AGM provides a better fit for app data than existing models and has the capability to predict the functions of apps.

• Both the word distributions and the topic distributions generated by AGM are more distinctive than those generated by the two baselines as the KL-divergence evaluation indicates.

• AGM can obtain better precision than the traditional methods in terms of the keyword extracted for the description of the app functions.

7 Conclusions

In this paper, we explored a novel approach of inferring the real functions of apps. We proposed a sophisticated probabilistic topic model named App Generative model (AGM) that captures different co-occurrences relations in the app corpus in a proper way. By integrating textual descriptions, app links, and the word burstiness phenomenon in app description, AGM is able to derive semantically coherent topics from the app corpus. We evaluated the proposed AGM via extensive experiments and the results showed that it significantly outperforms the state-of-art topic models. We believed that AGM can help the downstream app search engines to better capture the real functions of apps and thus can bring better app search services for the end users. Finally, our app feature extraction algorithm can effectively derive the precise app features.

References

[1] Liu S, Wang S, Zhu F, Zhang J, Krishnan R. HYDRA: Large-scale social identity linkage via heterogeneous behavior modeling. In *Proc. ACM SIGMOD*, June 2014, pp.51-62.

[2] Tong Y, Cao C C, Chen L. TCS: Efficient topic discovery over crowd-oriented service data. In *Proc. the 20th SIGKDD*, August 2014, pp.861-870.

[3] Baeza-Yates R, Jiang D, Silvestri F, Harrison B. Predicting the next app that you are going to use. In *Proc. the 8th WSDM*, February 2015, pp.285-294.

[4] She J, Tong Y, Chen L, Cao C C. Conflict-aware event-participant arrangement. In *Proc. the 31st ICDE*, April 2015, pp.735-746.

[5] She J, Tong Y, Chen L. Utility-aware social event-participant planning. In *Proc. ACM SIGMOD*, May 31-June 4, 2015, pp.1629-1643.

[6] Tong Y, Meng R, She J. On bottleneck-aware arrangement for event-based social networks. In *Proc. the 31st ICDE Workshops*, April 2015, pp.216-223.

[7] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2003, 3: 993-1022.

[8] Griffiths T L, Steyvers M. Finding scientific topics. *Proc. the National Academy of Sciences*, 2004, 101(Suppl.1): 5228-5235.

[9] Jo Y, Oh A H. Aspect and sentiment unification model for online review analysis. In *Proc. the 4th WSDM*, February 2011, pp.815-824.

[10] Sato I, Nakagawa H. Topic models with power-law using Pitman-Yor process. In *Proc. the 16th SIGKDD*, July 2010, pp.673-682.

[11] Wang C, Wang J, Xie X, Ma W Y. Mining geographic knowledge using location aware topic model. In *Proc. the 4th ACM Workshop on GIR*, November 2007, pp.65-70.

[12] Yin Z, Cao L, Han J, Zhai C, Huang T. Geographical topic discovery and comparison. In *Proc. the 20th WWW*, March 28-April 1, 2011, pp.247-256.

[13] Jiang D, Vosecky J, Leung K W T, Ng W. G-WSTD: A framework for geographic web search topic discovery. In *Proc. the 21st CIKM*, October 29-November 2, 2012, pp.1143-1152.

[14] Jiang D, Leung K W T, Ng W, Li H. Beyond click graph: Topic modeling for search engine query log analysis. In *Proc. the 18th DASFAA*, April 2013, pp.209-223.

[15] Sizov S. Geofolk: Latent spatial semantics in Web 2.0 social media. In *Proc. the 3rd WSDM*, February 2010, pp.281-290.

[16] Eisenstein J, O'Connor B, Smith N A, Xing E P. A latent variable model for geographic lexical variation. In *Proc. the EMNLP*, October 2010, pp.1277-1287.

[17] Jiang D, Leung K W T, Vosecky J, Ng W. Personalized query suggestion with diversity awareness. In *Proc. the 30th ICDE*, March 31-April 4, 2014, pp.400-411.

[18] Jiang D, Leung K W T, Ng W. Query intent mining with multiple dimensions of web search data. *World Wide Web*, 2015.

[19] Hao Q, Cai R, Wang C, Xiao R, Yang J M, Pang Y, Zhang L. Equip tourists with knowledge mined from travelogues. In *Proc. the 19th WWW*, April 2010, pp.401-410.

[20] Teh Y W. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. the 44th ACL*, July 2006, pp.985-992.

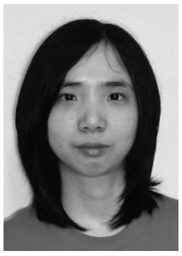
[21] El-Arini K. Dirichlet Processes: A Gentle Tutorial. 2008. https://www.cs.cmu.edu/∼kbe/dp_tutorial.pdf, Aug. 2015.

[22] Wallach H M. Structured topic models for language [Ph.D. Thesis]. Univ. Cambridge, 2008.

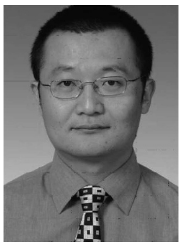
[23] Rosen-Zvi M, Griffiths T, Steyvers M, Smyth P. The author-topic model for authors and documents. In *Proc. the 20th UAI*, July 2004, pp.487-494.

[24] Xia H, Li J, Tang J, Moens M F. Plink-LDA: Using link as prior information in topic modeling. In *Proc. the 17th DASFAA*, April 2012, pp.213-227.

Yong-Xin Tong received his Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2014. He is currently an associate professor in the School of Computer Science and Engineering, Beihang University, Beijing. Before that, he served as a research assistant professor and a postdoctoral fellow at HKUST. He is a member of CCF, ACM, and IEEE. His research interests include crowdsourcing, uncertain data mining and management, and social network analysis.



Jieying She is currently a Ph.D. student at the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong. Her major research interest is managing event-based social networks. She is a student member of IEEE.



Lei Chen received his B.S. degree in computer science and engineering from Tianjin University, Tianjin, in 1994, M.A. degree from Asian Institute of Technology, Bangkok, Thailand, in 1997, and Ph.D. degree in computer science from the University of Waterloo, Canada, in 2005. He is currently an associate professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. So far, he has published over 200 conference and journal papers. He got the Best Paper Awards in DASFAA 2009 and 2010. He is PC Track chairs for SIGMOD 2014, VLDB 2014, ICDE 2012, CIKM 2012, SIGMM 2011. He has served as PC members for SIGMOD, VLDB, ICDE, SIGMM, and WWW. Currently, Prof. Chen is an associate editor-in-chief for IEEE Transactions on Knowledge and Data Engineering and serves on the editorial board of Distributed and Parallel Databases. He is a member of the VLDB endowment committee and the chairman of ACM SIGMOD China Chapter. His research interests include crowdsourcing over social media, social media analysis, probabilistic and uncertain databases, and privacy-preserved data publishing.

Appendix

We show some statistics of the 30 app categories in Google Play in Table A1, each of which consists of the corresponding occupancy, the average rating, the average number of users, and the average downloads of the corresponding category.

Table A1. Basic Statistics about Real App Categories

Category	Percentage	Ave. Rating	Ave. Number of Users	Ave. Last 30 Days Downloads
Brain & Puzzle	10.37	4.1	8 273	975 450
Arcade & Action	11.67	4.1	13 687	1 296 136
Communication	8.54	3.9	16 813	2 869 507
Tools	7.95	4.2	10 511	2 389 196
Music & Audio	3.24	4.2	15 126	1 878 414
News & Magazines	3.66	3.8	2 519	491 613
Entertainment	10.06	3.7	1 647	400 648
Personalization	8.03	4.1	3 796	778 657
Social	5.07	3.8	20 086	2 812 339
Sports Games	3.12	3.8	2 845	807 113
Casual	4.28	4.0	4 440	461 194
Travel & Local	1.80	3.8	31 461	6 557 724
Photography	3.83	4.0	4 681	1 478 874
Education	2.45	4.1	886	280 494
Media & Video	1.83	4.0	8 446	4 990 682
Productivity	3.55	4.1	11 021	1 459 246
Racing	0.95	3.9	10 824	1 450 213
Lifestyle	2.96	3.9	1 592	314 453
Books & Reference	2.06	4.3	244	111 285
Sports	0.73	4.1	652	1 447 067
Shopping	0.34	3.1	4 076	336 000
Comics	0.67	3.6	2 043	294 750
Cards & Casino	0.65	4.0	4 509	742 173
Weather	0.19	3.7	7 116	1 206 428
Transportation	0.17	3.3	19	6 051
Business	0.59	4.1	3 059	467 892
Libraries & Demo	0.45	4.2	2 605	417 843
Health & Fitness	0.48	4.2	789	107 955
Medical	0.11	4.0	590	1 087 500
Finance	0.11	2.3	2	2 634