This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

App Miscategorization Detection: A Case Study on Google Play

Didi Surian, Suranga Seneviratne, Aruna Seneviratne, and Sanjay Chawla

Abstract—An ongoing challenge in the rapidly evolving app market ecosystem is to maintain the integrity of app categories. At the time of registration, app developers have to select, what they believe, is the most appropriate category for their apps. Besides the inherent ambiguity of selecting the right category, the approach leaves open the possibility of misuse and potential gaming by the registrant. Periodically the app store will refine the list of categories available and potentially reassign the apps. However, it has been observed that the mismatch between the description of the app and the category it belongs to, continues to persist. Although some common mechanisms (*e.g.* a complaint-driven or manual checking) exist, they limit the response time to detect miscategorized apps and still open the challenge on categorization. We introduce FRAC+: (FR)amework for (A)pp (C)ategorization. FRAC+ has the following salient features: (i) it is based on a data-driven topic model and automatically suggests the categories appropriate for the app store, and (ii) it can detect miscategorizated apps. Extensive experiments attest to the performance of FRAC+. Experiments on GOOGLE Play shows that FRAC+'s topics are more aligned with GOOGLE's new categories and 0.35%-1.10% game apps are detected to be miscategorized.

Index Terms—App categorization, miscategorization detection, app market, von-Mises Fisher distribution, mixture model.

1 INTRODUCTION

App markets, where app developers can make their apps available to potential users, have been created by mobile device and platform manufacturers. The two dominant app markets, APPLE App Store and GOOGLE Play Store, each hosts over one million apps spread over 15+ categories [1], [2]. Now other players, including mobile operators, are also launching their own app markets. With the increasing popularity arises a concomitant downside that the app markets are becoming increasingly difficult to navigate and app categories harder to distinguish. As a result, app developers struggle to get visibility for their products. App search is not as advanced as web search and technology for "app search optimization" is still nascent.

An important factor that influences the visibility of an app is how it is categorized in the app market. In this context, *miscategorization* is one important problem that needs to be addressed by the app markets. The miscategorization problem is potentially introduced during the registration of an app in the app market. App developers are free to choose a category that they think is most suitable for their apps. However, the act of selecting a category for an app is inherently ambiguous and also opens up the possibility of deliberate gaming in order to avoid competition and improving the rank of the app [3]. The practice is attracting the attention of app market managers and is being discouraged and in some cases forbidden [4], [5].

Another reason for developers to register their apps under a less appropriate category is to avoid scrutiny. For example, the chances of a low quality personal health information app being able to remain in the app market without drawing significant attention to it, is not to categorize it under *Medical* category [6]. An example of miscategorization taken from GOOGLE Play Store is shown in Table 1 where a number puzzle app is categorized under the *Words* category.

TABLE 1
An Example of A Miscategorization in GOOGLE Play

App name	2048 **[7]**	2048 **[8]**
Registered Category	*Brain & Puzzle*	*Words*
App Description	Most addictive mobile version of 2048 game and almost perfect 2048 number puzzle game for Android! ...	Slide the screen, making the box move. Moving process, sliding in the same direction with the same...

An example of a miscategorization: *2048* is a popular *number puzzle game*[1]. Several implementations of this game are available in GOOGLE Play Store. This example shows a scenario where a developer has published one implementation under *Words* category, which is not relevant.

Miscategorization of apps has several implications: (i) it impairs the integrity of existing categories, (ii) it allows some app developers to get an unfair advantage over others [5], (iii) it makes auditing and ensuring quality/regulatory control more difficult [6] and (iv) it might mislead users and entice them to pay for apps that do not provide the expected utility [5]. Thus, it is important to have a robust

- *D. Surian is with Macquarie University and the work was done while the author was with the School of Information Technologies, the University of Sydney, NSW, Australia, 2008 and National ICT of Australia, NSW, Australia, 2015.*
  Email: didi.surian@mq.edu.au
- *S. Seneviratne and A. Seneviratne are with the University of New South Wales, NSW, Australia, 2052 and Data61, CSIRO, NSW, Australia, 2015.*
  Email: {Suranga.Seneviratne,Aruna.Seneviratne}@data61.csiro.au
- *S. Chawla is with Qatar Computing Research Institute, HBKU, Doha, Qatar.*
  Email: schawla@qf.org.qa

[1] https://en.wikipedia.org/wiki/2048_(video_game)

categorization and miscategorization detection system in the app markets to protect end-users and maintain a healthy competitive ecosystem.

According to existing reports [9], [10], the current methods of policy violation detection (including miscategorization detection) are either manual or complaint-driven. For example, APPLE appears to manually check the app metadata and decide whether it is allowed to be published in the market or not. While this method works and identifies miscategorized apps correctly, it will increase the length of time to get an approval for an app. On the other hand, GOOGLE seems to use a complaint-driven strategy. An app will be removed from the GOOGLE Play Store if there are many complaints from the users. Apart from this information, most companies would use their own proprietary approaches to deal with miscategorization problems. In this paper, we propose an automated general method to perform app categorization and detect miscategorization. We compared our method with baseline methods and as well as against human judgement (Section 5 and 6). We make the following contributions.

1) We present FRAC+, (FR)amework for (A)pp (C)ategorization based on probabilistic modeling to categorize and detect miscategorized apps. FRAC+ models app descriptions as *normalized vectors* in the space of words. We subsequently show that FRAC+ is more accurate than other generic models such as LDA which are based on *word-count*.
2) We performed extensive experiments on synthetic, semi-synthetic and real data sets. On synthetic data sets, we show that FRAC+ outperforms baseline methods in detecting the correct number of categories even though there exists a category that has very low number of apps in contrast to other categories. On semi-synthetic and real data sets, we show that FRAC+ outperforms baseline methods on categorization and detecting miscategorization tasks.
3) We applied FRAC+ to GOOGLE Play Store data and show that among the game app categories there exist approximately 0.35%–1.10% miscategorizations, under the most conservative assumption and can be as high as 3.32%–11.08%. We also propose a new categorization and evaluate it against the new GOOGLE's game app categories.

2 RELATED WORK

To the best of our knowledge the problem of app miscategorization has not been addressed before in the research literature. We summarize research in app classification, detection of malicious apps, fraud in app rankings, classification and topic modeling problems.

App classification. Several work looked into the possibility of using topic models to classify apps into meaningful categories. Vakulenko *et al.* [11] applied LDA to a set of 600,000 English app descriptions obtained from Apple App Store. Authors generated the same number of topics as that is available in Apple App Store and compared the overlap between the LDA topics and the existing app categories in Apple App Store assuming existing categories are manually curated by domain experts. Results showed that LDA identifies some new topics in likes of radio stations, timers, calculators, quizzes, banking, and sound effects that are absent in the original app store categorization. Al-Subaihin *et al.* [12] proposed an app classification methodology based on agglomerative hierarchical clustering and showed that existing categorizations can be further improved. Berardi *et al.* [13] used a SVM classifier to classify apps into user defined categories using textual features generated from app descriptions and current category of the app as defined in Google Play Store or Apple App Store. Zhu *et al.* [14] highlighted that it is a non-trivial task to effectively classify mobile apps due to the limited contextual information available for the analysis. Zhu *et al.* [15] and Lulu *et al.* [16] looked into the possibility of improving app classification quality by adding external information obtained by web search.

Malicious app behavior. Multiple work studied about malicious app behavior [17], [18], [19], [20], [21]. Gorla *et al.* [17] proposed CHABADA framework that clustered apps based on app descriptions using the combination of a LDA topic model and k-means clustering algorithm to identify apps whose API usage is anomalous within the respective topic cluster using one-class Support Vector Machines. An example of a malicious case is that of a *weather* app, which makes API calls to send SMSs. Wei *et al.* [18] proposed a multi-layer system to profile apps. The proposed scheme uses *static information* such as requested permissions, *user generated events* such as touch events, *invoked system calls,* and *network traffic* to profile apps. Then each layer of an app is assigned an intensity: high, medium, or low. Apps showing sudden changes in intensity levels in some layers were considered as potentially malicious. Ma *et al.* [22] used a semi-supervised approach to the same problem and show that the new method outperforms CHABADA. Sanz *et al.* [21] and Shabtai *et al.* [23] used app metadata, requested permissions and behavior features to classify malicious apps using machine learning techniques.

Ranking fraud and app review mining. App ranking fraud detection was explored recently. Zhu *et al.* [24] used statistical hypothesis testing on app rankings to identify developers who are fraudulently trying to improve the ranking of their apps. The features used were, for example, how long an app is in the top charts and its average rating when it is in the top chart compared to its previous ratings. Chandy *et al.* [25] developed a model to detect fake ratings using rating-related features, such as the average rating of an app and the users' average ratings. However, the intention of ranking fraud is only one scenario in miscategorization and generally, ranking fraud happens after the publication of the apps. Fu *et al.* [26] identified the major factors resulting lower app ratings by mining app reviews and ratings using LDA and linear regression. Liu *et al.* [27] used the similarity between app reviews to identify relationships between apps such as functional similarity and complementary or dependent app pairs.

Classification and novelty detection problems. The prob-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

lem of app categorization is closely related to the classification problem. In the classification problem, one aims to identify a category (or a class) membership of an observation from a set of categories. The classification is done by a classifier that groups observations which share similar properties under the same category/class. Another aspect of the classification problem is novelty detection. In novelty detection, one aims to recognize observations that have different patterns from other observations[2]. Several methods have been used as the de-facto approaches in classification and novelty detection including *clustering-based*, *domain-based* methods and *generative models.*

Clustering-based methods group a set of observation points to several groups/clusters. The most common method to group a set of n observation points to k clusters is k-means clustering algorithm. Each cluster is represented by a centroid or a cluster center, which is a mean of all nearest observation points within that cluster. Observation points that have large distance to the cluster center in each cluster are considered to be anomalous. The k-means is widely adopted in many different applications, for example, Zhou *et al.* [28] and Yoon *et al.* [29] used k-means to cluster data and detect outliers.

Domain-based methods separate observations based on a boundary learned from the training data. The de-facto approach to domain-based novelty detection has been one-class Support Vector Machines (OC-SVM) [30]. OC-SVM defines a hyperplane that maximizes the separating margin between classes. Observation points that lie near the boundary are called support vectors. The bigger distance of an observation point to the defined hyperplane, the further an observation from the commonly observed behavior. OC-SVM has also been studied extensively for various applications. Manevitz *et al.* [31] reviewed OC-SVM for document classification.

Generative models assume that normal observations are generated from some underlying distributions. Observations that do not follow the general patterns from the underlying distributions are identified as outliers [32]. A *state-of-the-art* generative model to model text documents is Latent Dirichlet Allocation (LDA) [33]. LDA is mainly used to infer a set of topics from a corpus of documents. We discuss LDA in the next paragraphs.

Topic modeling. There is a large body of work that uses topic models in many different application domains. Blei *et al.* [33] proposed Latent Dirichlet Allocation (LDA), a generative topic model based on Bayesian statistics. LDA has been widely adopted [34], [35]. Ahmed *et al.* [34] used LDA to capture users' long and short term interests based on their online activities. Ramage *et al.* [35] introduced a topic model that constrains LDA by defining a one-to-one correspondence between topics from LDA and user tags. The probabilistic modeling of normalized vector data has recently given rise to the use of the von Mises-Fisher (vMF) distribution for modeling directional data. For example, Banerjee *et al.* [36] introduced a mixture of vMF distributions to model data. Surian *et al.* [37] proposed a topic model using vMF distribution to detect outlier collaborations in DBLP bibliography data. FRAC+ also uses the vMF distribution, but integrates existing app categories into the model. Other studies about topic modeling have also included the information from a pre-existing taxonomy in data for classification of documents [38], [39]. Some studies have discussed new categorization system based on the existing ones [40], [41], [42]. Topic modeling has also been studied in app categorization recently. For instance, Vakulenko *et al.* [11] used LDA to categorize iTunes apps.

[2]We consider observations under the same category share similar properties. For example, apps registered in *Brain & Puzzle* category should ideally be distinct from apps in *Racing*.

3 Background

Inferring categories from data can be addressed using clustering and topic modeling. For example, a popular approach, based on topic modeling, is the use of count statistics in conjunction with the Latent Dirichlet Allocation (LDA) probabilistic model [33]. In LDA we are given a corpus of documents and the objective is to infer topics such that each document is a distribution in the topic space and each topic is a distribution on the word space. Since the number of topics is usually much smaller than the number of words, LDA can be seen as a form of dimensionality reduction in the topic space. The topics can then be treated as categories. However, LDA is tightly coupled with a viewpoint where each data instance is associated with a feature vector consisting of discrete counts of words. Discrete counts can be naturally modeled using the multinomial distribution, which is a conjugate prior of the Dirichlet distribution and thus the name LDA. The use of conjugate priors substantially simplifies the computational task of estimating the parameters of the model.

Research in information retrieval and text mining has shown that machine learning techniques yield better performance when they are applied to documents (*e.g.*, app descriptions) which are normalized. A common normalization for documents is to use tf-idf (term frequency inverse document frequency) [43]. The tf-idf normalization gives higher weight to terms in a document that characterize the document and lower weight to those terms that occur across many documents. Just as the multinomial distribution is closely associated with count data, the lesser known von Mises-Fisher (vMF) distribution plays an analogous role for normalized vectors [44]. Before we give further details about the vMF distribution, we illustrate the impact of count and normalized data with the help of a simple example.

3.1 Count vs. Normalized Frequency

Assume that there are eight game apps. Each game app is represented by a feature vector extracted from the app description. Further assume that the vocabulary consists of three words: "fight", "war" and "puzzle". Table 2 shows the word frequencies for the example.

Assume that we want to cluster the eight game apps shown in Table 2. Table 2 shows that only `Games 7 & 8` that have the word "puzzle", while `Games 1–6` share the words "fight" and "war" with various different proportions. The distribution of words (Game 1–3, 4–6, 7–8) suggests that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
Word Frequencies For The Example Apps

Game App	*fight*	*war*	*puzzle*
Game 1	8	3	0
Game 2	9	2	0
Game 3	10	3	0
Game 4	2	10	0
Game 5	2	7	0
Game 6	2	9	0
Game 7	0	0	1
Game 8	0	0	2

the apps could be clustered into three clusters. However, notice that the words from Games 1–6 ("fight", "war") are semantically closed, thus intuitively `Games 1-6` should be in the same cluster and `Games 7 & 8` should be clustered separately. We aim to have apps with similar themes in a same group.

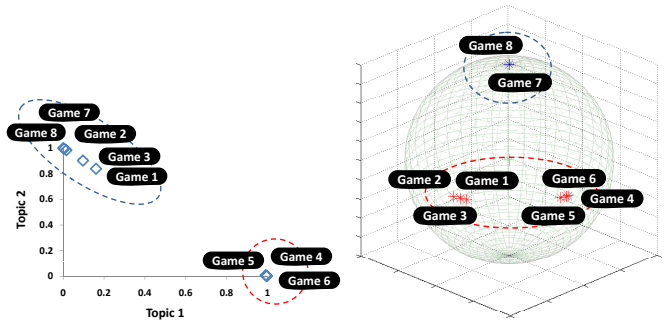
A natural approach to group those eight game apps is to use Latent Dirichlet Allocation (LDA) [33] which performs an analysis in the topic space. However, LDA introduces some drawbacks because it is mainly affected by the word count statistics. We can observe that based on their word counts, `Games 1-3` are more about *fight* and `Games 4-6` are more about *war*. Because both `Games 7 & 8` have only fewer words than `Games 1-6` do, `Games 7 & 8` will have to choose to join either to `Games 1-3` or `Games 4-6`, which gives an undesirable clustering result (notice that `Games 7 & 8` do not share any common words with either `Games 1-6` or `Games 7 & 8`). Figure 1(a) shows topic proportions and the resulting clusters from LDA in a 2-D plot.

On the other hand, if we represent the feature vectors from the game apps as unit vectors on a sphere shown in Figure 1(b), `Games 7 & 8` are well separated from `Games 1-6`. Using our proposed model described later in details, `Games 1-6` and `Games 7 & 8` are grouped separately. This result follows our initial intuition[3]

For quantitative comparison, we use silhouette to assess the quality of clusters. The silhouette is a method to interpret how similar an object is to its cluster [45]. We give more details about silhouette later in this work. The average silhouette values for our model and LDA is **0.77** and 0.43 respectively (the value is close to 1 if each member is well matched to its own cluster.), which suggests that if the eight game apps are grouped into two clusters, then `Games 1-6` and `Games 7 & 8` are more similar among themselves respectively.

We also experimented on REUTERS-21578 [33], a larger and real data set which contains a number of labelled articles. We categorized the articles into: 'earn' (3,776 articles), 'grain' (574 articles), and 'not earn and not grain' (8,772 articles). Standard pre-processing techniques were performed (refer to Section 5.2 for feature construction) and we set the number of topics for both LDA and our model to 3. The resulted NMI scores for FRAC+ and LDA are **0.56** and 0.30 respectively, with higher score represents better results.

[3]With number of topics = 3, both models give same clustering result. However, as described previously, two clusters is preferable.



(a) (b)

Fig. 1: If we set the number of topics = 2: (a) topic proportions from LDA in a 2-D plot (b) representation using unit vectors on a sphere which correctly splits the games into two clusters. See text for details.

4 FRAC+: A FRAMEWORK FOR APP CATEGORIZATION AND MISCATEGORIZATION DETECTION

4.1 The vMF Distribution For Modeling App Data

The core of FRAC+ is a topic model using vMF distribution. The vMF distribution is like the Gaussian distribution on a sphere [44]. The probability density function of the vMF distribution for a d-dimensional unit random vector $\boldsymbol{X}$ is:

$$f(\boldsymbol{X}|\boldsymbol{\mu},\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa)} e^{\kappa \boldsymbol{\mu}^T \boldsymbol{X}}$$

where $\|\boldsymbol{\mu}\| = 1$ ($\boldsymbol{\mu}$ is scaled to be of unit length), $\kappa \geq 0$, $d \geq 2$, and $I_r(\cdot)$ represents the modified Bessel function of the first kind of order r. The parameters $\boldsymbol{\mu}$ and κ are called the *mean direction* and *concentration parameter* respectively. The parameter κ characterizes how strongly the unit vectors drawn based on probability density function $f(\boldsymbol{X}|\boldsymbol{\mu},\kappa)$ are concentrated about the *mean direction* $\boldsymbol{\mu}$. Specifically, if $\kappa = 0$, the distribution is uniform and, if $\kappa \to \infty$, the distribution tends to concentrate around the mean vector $\boldsymbol{\mu}$. Banerjee *et al.* [36] have discussed an approximation to get κ value as in Equation 1.

$$\kappa \approx \frac{\bar{\mathbf{r}} d - \bar{\mathbf{r}}^3}{1 - \bar{\mathbf{r}}^2} \tag{1}$$

where $\bar{\mathbf{r}} = \frac{\|\mathbf{r}\|}{N}$, $\mathbf{r} = \sum_n \boldsymbol{x}_n$. $\bar{\mathbf{r}}$ is called the mean resultant length and N is the number of observations. Figure 2 shows 1,000 points sampled from a vMF distribution with $d = 3$, $\kappa = 150$ (blue points - right side of sphere) and $\kappa = 10$ (red points - left side of sphere). The *mean directions* are shown as solid and dashed lines from the centre of sphere. More details about directional statistics and the vMF distribution can be found in [36], [44].

Notice from Figure 2 that larger value of κ will make the distribution to concentrate on one density. Each point shown in Figure 2 is represented by a unit vector that defines its position on the sphere and points with different color represent different clusters/categories those points belong to.

In the application of app categorization, we can use μ to represent a category (topic) and κ to show how concentrated are the apps in a category. As the proportion of miscategorized apps in a category is relatively small in a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

category, we can calculate κ first for the existing category and aim to cluster the apps based on their features. FRAC+ uses this concept to categorize and to find miscategorized apps. We make an analogy as follows. A document consists of a number of words, similarly a category in app market consists of a number of apps and each app is represented as a unit vector. This analogy is shown in Table 3 with the description whether the information is available or not.

TABLE 3
Analogy: Generic Topic Modeling Concepts And App Data Set

Generic Concepts	App Data Set	Status
Document	App category	Observed
Words	App unit vectors	Observed
Topics	Inferred categories	Latent

4.2 Generative Process

We now describe a generative process which we assume instantiates the observed data. Bold-faced variables, *e.g.*, $\boldsymbol{\mu}$, $\boldsymbol{X}$, $\boldsymbol{\theta}$, represent vectors. Let $\boldsymbol{X_{mn}}$ denote the feature vector for app n in category m. Algorithm 1 shows the data generative process for FRAC+.

Algorithm 1 Generative process of FRAC+

1: **for** every category $m \in M$ **do**
2: Compute κ_m (Eq. 1)
3: Sample topic proportions $\boldsymbol{\theta_m} \sim$ Dirichlet(.$\mid \alpha$)
4: **for** every app $n \in N_m$ **do**
5: Choose a topic $Z_{mn} \sim$ Multinomial(.$\mid \boldsymbol{\theta_m}$)
6: Generate a unit feature vector $\boldsymbol{X_{mn}} \sim$ vMF(.$\mid \boldsymbol{\mu}, \kappa_m$)
7: **end for**
8: **end for**

Note, that the only variable that is visible (available) is the set of app descriptions $\{\boldsymbol{X_{mn}}\}$ in each category. Our objective now is to learn the latent variables and parameters $\{Z_{mn}, \boldsymbol{\theta_m}, \boldsymbol{\mu_k}, \kappa_m\}$. A key innovation of the model is the coupling between $\boldsymbol{\mu_k}$ (which represents a new category) and κ_m from existing category. The coupling allows us to integrate the existing with the new inferred categories. Due to space constraint, we invite interested readers to our supplementary material for more details on how we performed the inference and learning process for the topic model. Figure 3 shows the graphical model of our topic model. A graphical model uses a *plate* (rectangle) to represent a repetition of a variable, instead of representing each repeated variable individually, *e.g.*, a variable that represents *w*ords on a document could be drawn on a plate for words. A circle on a plate represents the variable itself. An arrow represents a dependency assumption. For example $\boldsymbol{\theta_m}$ for each category depends upon the Dirichlet parameter α. The dark shaded circle represents observed variables and the unshaded circle represents latent variables. We introduce the grey shaded circle for variables that can be directly computed from the data set (*e.g.*, κ_m).

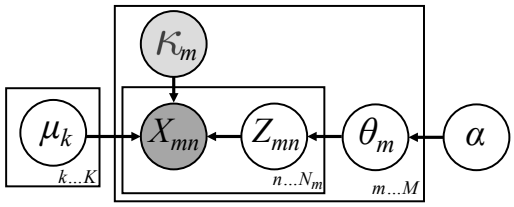


Fig. 3: A graphical model for representing the data generating process of our topic model.

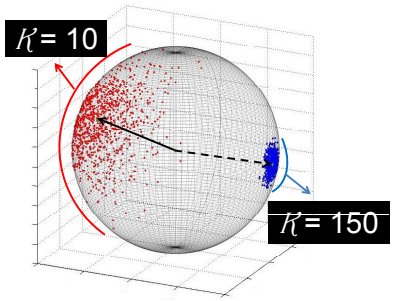


Fig. 2: Points on a sphere (d = 3) drawn from a vMF distribution: κ = 150 (blue points - right side of the sphere) & κ = 10 (red points - left side of the sphere).

4.3 FRAC+ Overall Framework

We present FRAC+ as a complete framework in Figure 4. FRAC+ performs two main processes in order to answer the following key research questions: 1) finding the optimal number of categories, and 2) detecting miscategorized apps. We present the details in the following paragraphs.

`Process 1`: We ignore the app category information and treat all apps as they are in one big category ($M = 1$). To find the optimal number of topics K, we run our topic model for various number of topics K and compute the *silhouette value* [46]. The silhouette value is a metric to determine how well a point is matched to its cluster by measuring the difference between inter-cluster and intra-cluster (dis)similarity. Assume that we have clustered a data set into K clusters. Each cluster is labeled as k, $k \in \{1,\ldots,K\}$. Let n_k represents a point (observation) n in cluster k, then the silhouette value is calculated as:

$$s(n_k) = \frac{b(n_k) - a(n_k)}{max\{a(n_k), b(n_k)\}}$$

where $a(n_k)$ represents the average dissimilarity of n_k with all other points in cluster k, and, $b(n_k)$ represents the lowest average dissimilarity of n_k to any other clusters k', where $k' \in \{1,\ldots,K\}$, $k' \neq k$. The dissimilarity could be computed using, *e.g.*, distance measure. We use *cosine* similarity in this work. Note that dissimilarity is equal to (1-similarity). Higher value of silhouette means that the points are more similar to the rest points in their cluster rather than in neighboring clusters.

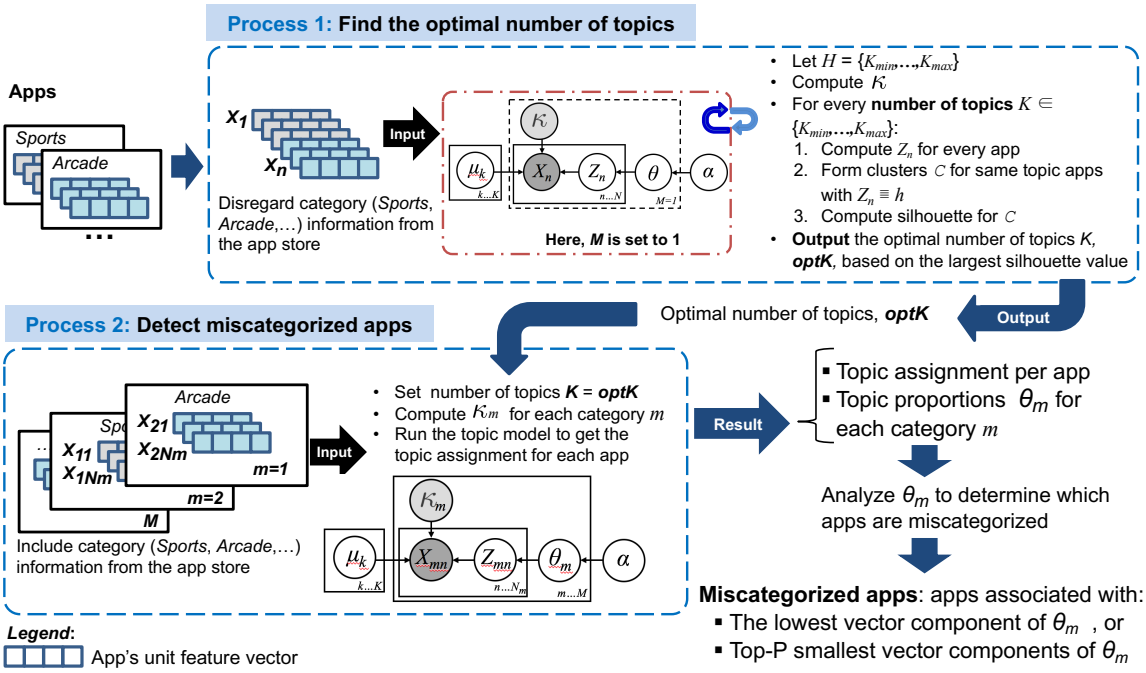


Fig. 4: Two main processes in FRAC+: **`Process 1.`** Detect the optimal number of topics and **`Process 2.`** Detect miscategorization - which is achieved by running again the topic model using the optimal number of topics from `Process 1` and the app category information. See text for details.

We take the average silhouette[4] values when determining the optimal number of topics K. We choose K with the highest silhouette value and label it as $optK$.

`Process 2`: To find miscategorized apps, we run again our topic model with $K = optK$. Contrary to `Process 1`, here we include the app category information (recall the analogy in Table 3) to our topic model. After we run the topic model, each app is associated with a topic $k \in \{1,\ldots,K\}$. Then for each category m, we examine its topic proportions, which is the vector $\boldsymbol{\theta_m}$. In order to detect miscategorized apps, we use the following approach.

A category in GOOGLE Play Store reflects the topic of that category. For example, ideally all apps registered in *Racing* are about racing (*e.g.*, Real Racing game app[5,7]). Due to some reasons that we described in Section 1, some apps (*e.g.*, Alpha Zoo game app[6,7]) with different topics could also be listed under *Racing*. While it is particularly true that all apps with different topics should be marked as miscategorized apps, in this work we are more interested in analyzing the apps associated with the smallest value component of the vector $\boldsymbol{\theta_m}$. Therefore, all apps associated with the smallest vector component are labeled as being miscategorized. A more relaxed condition would be to include apps associated with top-P smallest vector components. One (*i.e.*, the app store managers) could determine to use the former or latter option based on their requirements. In Section 6, we show our case study on GOOGLE Play Store data set using these two options.

5 EXPERIMENTS

We conducted our experiments[8] on a machine with Intel® Core(TM) Duo CPU T6400 @2.00 GHz, 1.75 GB of RAM. Our experiments are designed to answer the following questions:

1) Can FRAC+ be used to determine the *optimal* number of categories?
2) Can the model be used to identify miscategorized apps?
3) How does the topic model algorithm scale to a large number of apps and categories?
4) Does the framework generate categories which are semantically coherent, *i.e.*, the categories can be used by the app store to catalog its existing suite of apps?

5.1 Baseline Methods

The key novelty in FRAC+ is a new topic model that is based on directional distributions to generate topics from the app descriptions. We compare the performance of FRAC+ to Latent Dirichlet Allocation (LDA) [33], *k*-means++ [47], multi-class SVM and one-class SVM.

We use the implementation of LDA from Stanford Topic Modeling toolbox v0.4.0 [48]. For all experiments in this

[4]For simplicity, henceforth we will refer the *average silhouette* as *silhouette*.

[5]https://play.google.com/store/apps/details?id=com.ea.games.r3_na.

[6]https://play.google.com/store/apps/details?id=com.funbox.game.alphaZoo.

[7]As of February 2015.

[8]Our codes and data sets are available at https://sites.google.com/site/didisurian/home/codes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

work, we set both LDA term and topic smoothing parameters to 0.01 so the distribution will be concentrated in a few components. We use the implementation of collapsed variational Bayes approximation with 1,000 iterations. For SVM, we use LIBSVM 3.18 [49] with RBF kernel and select the parameter γ from $\{10^{-3}, 10^{-2}, 10^{-1}\}$ using 10-fold cross-validation. We train one-class SVM for each category. We assign an app to a topic with the largest proportion from the probability vector for the baseline methods that assign a probability to an app.

5.2 Data Sets

We used synthetic, semi-synthetic and real data sets to perform the experiments.

Synthetic Data (SD): For Synthetic Data, we create a scenario where there exist a small number of apps that have different topic than the others. We inject some noise to the data and included category information. The aim is to attest our `Process 1` (Figure 4) in order to detect the correct number of topics. The data set was generated as follows. We first fixed the number of categories M and the number of topics K. We associated each topic $k \in \{1,\ldots,K\}$ with a unique set of words, say, w_k. Let V represents a total vocabulary, so $|V|$ (total unique words) is $K \times w_k$. Let s_k^m be a set of synthetic apps in category m that contains words randomly sampled from w_k. In other words, each synthetic app has a feature vector where each entry represents how many times a word appears in each synthetic app. $s_{k=1}^m$ and $s_{k=2}^m$ also share some random common words (c.f. the scenario in Section 3). We set $M = 5$ and for each category $m \in \{1,\ldots,M\}$, we set $K = 3$, where $|s_{k=1}^m| = |s_{k=2}^m| = 150$, $|s_{k=3}^m| = 3$ (1% of $|s_{k=1}^m| + |s_{k=2}^m|$), and $|w_k| = 3$. There were 1,515 apps at the end of this process. We then picked randomly 50% these apps and injected random numbers to the feature vectors in order to represent some noise.

Real Data (RD): We used a subset of data set collected from GOOGLE Play Store that was used in our previous work [50]. In specific, we used: *app name, category* and *description*. The data set was collected up to March 31st, 2014[9]. For this work, we focus only on game apps as it is the most popular app type in app markets [52], [53]. There are six categories for game apps: *Arcade & Action*, *Brain & Puzzle*, *Casual*, *Racing*, *Cards & Casino*, and *Sports* and 48,663 game apps in our data set. We only included game apps that have at least 500,000 downloads and have more than 150 words in the description. Among the game apps in our data set, there are 40,180 apps having less than 500,000 downloads, 4,358 game apps that do not use English in the descriptions and 29,385 game apps have less than 150 words in the descriptions. We filtered them out from our data set. We used standard text mining techniques to process the app descriptions, *i.e.*, the removal of stop words, symbols and punctuations, and word stemming. We only included words (terms) that appear in at least 200 apps. After filtering and processing, there are 5,546 game apps and 672 unique words in the vocabulary. Each app is represented by a feature vector with dimension equal to the size of vocabulary and the position in the feature vector represents a word in the vocabulary. Each entry in the feature vector is a tf-idf value for the respective word. Figure 5 shows the distribution of words. Figure 6 shows the distribution of game apps in each category. For both FRAC+ and multi-class SVM, we reduced the feature vector dimension to 35 using PCA and scaled the feature vector to be of unit length for FRAC+. It had been shown previously [33] that SVM performs better on the reduced dimension of data set[10]. We normalized the feature vectors to form unit feature vectors for FRAC+ as we have described in Section 4.1. Figure 7 shows the κ values for each category m. We set α to 1 to represent a non-informative prior and let the model to learn the best value during the iteration [54], [55].

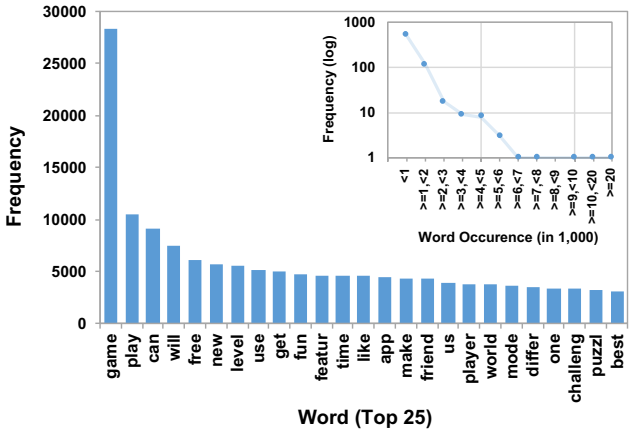


Fig. 5: Distribution of top-25 words (in a stemmed form). The inset figure shows the distribution of all words in the dictionary.

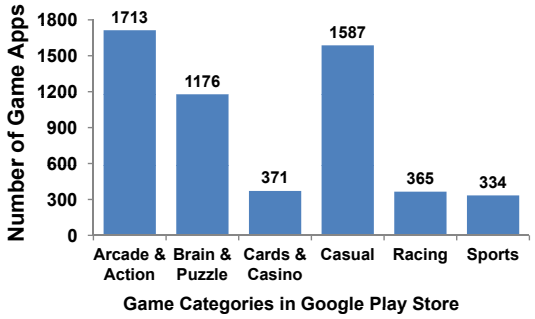


Fig. 6: Number of game apps per category in GOOGLE Play Store.

Semi-Synthetic Data (SSD): We created a semi-synthetic data by taking a subset of real data. Specifically, we took top-1,000 game apps based on the number of downloads and average ratings as our SSD. We then created new subsets by taking randomly 200, 400, 600, 800 and 1,000 apps from the data. To simulate miscategorization in each subset, we took randomly a number of apps from the two largest

[9]Before March 2014, GOOGLE Play Store had 30 app categories with 6 of them belonging to game apps. Later the number of categories in game apps was extended to 18 [51].

[10]Our initial experiments showed the same results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

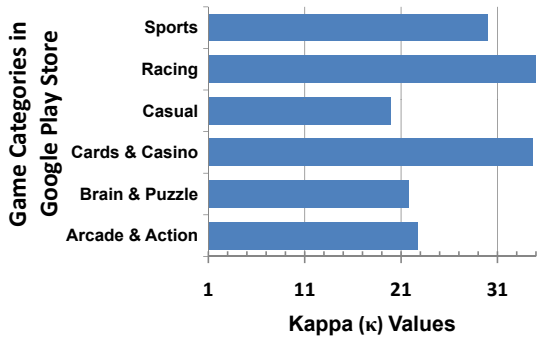


Fig. 7: κ value for each game category. Notice that category *Casual* has the lowest κ value, meaning there are many apps with different topics under this category, while game apps in *Racing* have more focused topic (larger κ value).

categories[11]. We then changed the categories one to another from the selected apps and swapped them. We recorded these apps as the miscategorized apps.

The following sections will try to answer our research questions: 1, 2 and 3 respectively using SD and SSD. To answer our research question 4, we present a case study on GOOGLE Play Store using RD and present the results in Section 6.

5.3 Finding Number of Optimal Topics

We performed experiments on SD to evaluate if we could infer the correct number of topics from the data sets. Recall that in SD we have set the number of topics $K = 3$. We treated the data set without including the information which categories (M) they originally belong to. As described in Section 4.3, silhouette is used to measure how well an object matches to its cluster. We varied the number of topics from 2 to 5 and computed the silhouette values. Given a number of clusters, the silhouette will be high if the method performs well in clustering the data. Specifically, this experiment refers to `Process 1` described previously in Section 4.3. As shown in Table 4, FRAC+ attains the highest value for $K = 3$. The other baseline methods attain their highest values on different number of topics with the LDA selecting $K = 4$ and the *k*-means++ selecting $K = 2$.

TABLE 4
Finding Optimal Number of Topics Based on The Silhouette Value

Number of topics	2	**3**	4	5
FRAC+	0.7209	†**0.7376**	0.7334	-0.1260
LDA	0.3457	0.7338	**0.7343**	0.0528
k-means++	**0.7209**	0.6630	0.570	0.5911

†Based on the silhouette value, FRAC+ is able to detect the correct number of topics (*i.e.*, 3) on semi-synthetic data set.

[11]We keep the term "category" for the real (and semi-synthetic) data set referring to GOOGLE Play Store categories.

5.4 Miscategorization Detection

We performed experiments on SSD to evaluate the performance in detecting miscategorization. Recall that for SSD that we have several subsets: 200, 400, 600, 800 and 1,000. For each subset, we ran FRAC+, took apps that belong to the top-5 lowest topic vector components and analyzed if these apps were the ones that we had changed the categories. We computed precision to see the performance of FRAC+ and the baseline methods in recognizing the miscategorized apps. If A is the set of miscategorized apps and B is the set of apps identified to be miscategorized, then the precision is expressed as:

$$\text{precision} = \frac{|A \cap B|}{|B|}$$

Note that as SSD was formed from the real data, therefore SSD *may already have* some miscategorized apps that we did not know in advance. Furthermore, the SSD construction process may also create a *reversing effect* where miscategorized apps are now in the correct categories. We did not control these two possible situations and only focused on identifying miscategorized apps that we have altered the categories. Table 5 shows that FRAC+ outperforms all baseline methods in most subsets, especially in the larger subsets, in finding the miscategorized apps in SSD. For example, FRAC+ constantly outperforms all baseline methods in 600, 800 and 1,000-subsets with 8% of apps were miscategorized.

5.5 Scalability

To test how FRAC+ scales, we varied the number of synthetic apps, number of categories, and number of topics while fixing the dimensionality of the vocabulary at 500 and 1,000. Figure 8 shows that FRAC+ scales in a near linear fashion. It takes under 30 seconds to process 8,000 synthetic apps with dimension of vocabulary is 1,000.

6 Case Study on Google Play Store Data

We applied FRAC+ to GOOGLE Play Store data described in Section 5.2. We present our qualitative and quantitative analysis in the following sections.

6.1 App categories according to FRAC+

We first evaluated the optimal number of topics by evaluating the silhouette values from K_{min}=4 to K_{max}=30. We chose this thresholds heuristically to best represent a possible number of categories in the app market. Using `Process 1` described in Section 4.3, we found that the "optimal" number of topics, $optK$, is 18 (Figure 9). If we treat $optK$ as the number of new categories, then the distribution of game apps is as shown in Figure 10.

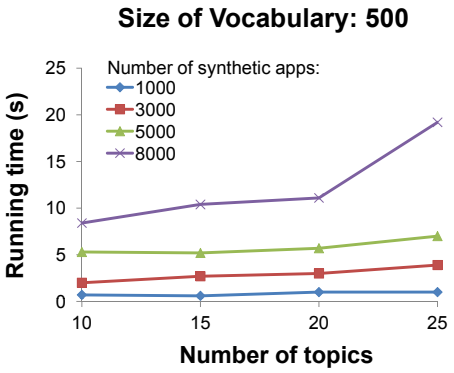
We then qualitatively analyzed the new topics returned by FRAC+. We tried to give meaningful names for the new topics by observing the word distribution in each topic. For example, we found that *Topic 4* and *Topic 13* correspond to *Card* and *Casino* games respectively. Moreover *Topic 8* represents *Racing* games.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

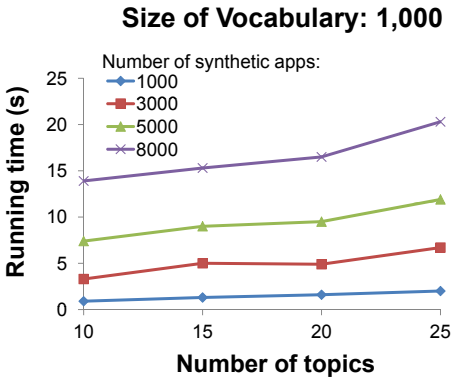
TABLE 5
Precision Scores in Finding Miscategorized Apps From SSD

Subset	Miscategorized	FRAC+	*k*-means++	LDA	M-SVM	OC-SVM
200	8%	0	0	0	**0.4**	0
	16%	**0.8**	0.2	0.2	0.67	0
400	8%	0	0.2	0	**0.33**	0
	16%	0	0.4	0.5	**0.67**	0
600	8%	**0.33**	0	0	0.2	0
	16%	0.4	0.5	**0.6**	0.4	0.4
800	8%	**0.6**	0.4	0.5	0	0
	15%	**0.5**	0.4	0.33	0.4	0.2
1,000	8%	**0.4**	0	0.25	0.2	0
	15%	**1**	0.5	0.4	0.2	0.4

Best (round-up) values are highlighted. M-SVM: Multi-class SVM, OC-SVM: One-class SVM.



(a)



(b)

Fig. 8: Scalability on synthetic data sets of sizes 1,000 to 8,000 and 10 to 25 topics. (a) Running time of vocabulary size 500, (b) running time of vocabulary size 1,000.

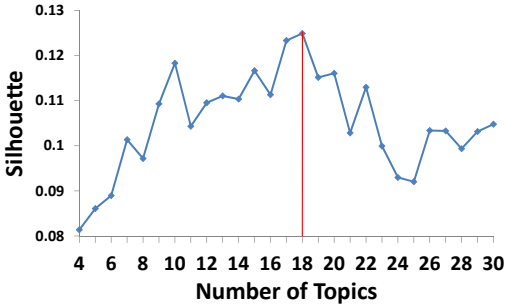


Fig. 9: Using `Process 1` described in Section 4.3, we found that the highest silhouette value is reached at $K = 18$ (the optimal number of topics for real data set)

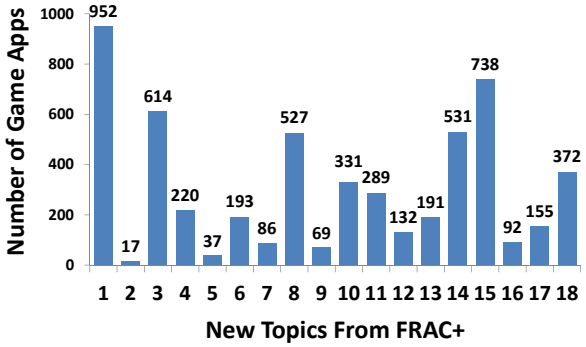


Fig. 10: Distribution of game apps returned by FRAC+ when K is 18.

Figures 11(a), 11(b), and 11(c) show the word clouds for *Topics 4, 13,* and *8* respectively. These figures show that the categorization by FRAC+ is semantically coherent.

Figure 12(a) shows the "heat map" between the topics derived from FRAC+ and GOOGLE's new categories. The color intensity represents the percentage of game apps that belong to FRAC+'s topic (row) and GOOGLE's category (column). As Figure 12(a) shows, there is a significant overlap between GOOGLE's new app categories and the topics from FRAC+. For example, (FRAC+)*Topic 1: Casual–Casual* (GOOGLE), (FRAC+)*Topic 3: Puzzle–Puzzle* (GOOGLE), and (FRAC+)*Topic 13: Casino–Casino* (GOOGLE) *etc.*

Altogether **nine** topics from FRAC+ have high intensity points along the diagonal of the heat map indicating those topics have corresponding similar topics in GOOGLE's new categories. Some topics from FRAC+ have high density points in multiple GOOGLE topics and vice versa. This is possible as some topics can be overlapping with others. For example, *Topic 5: Zombie* (FRAC+) has two high intensity points at *Action* and *Arcade* (GOOGLE). Figure 12(b) shows the "heat map" between the topics from LDA and GOOGLE's new categories. The heat map also shows high intensity points along the diagonal, indicating some topics from LDA also corresponds to GOOGLE's topics. However, in LDA there are only **seven** topics strongly corresponding

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

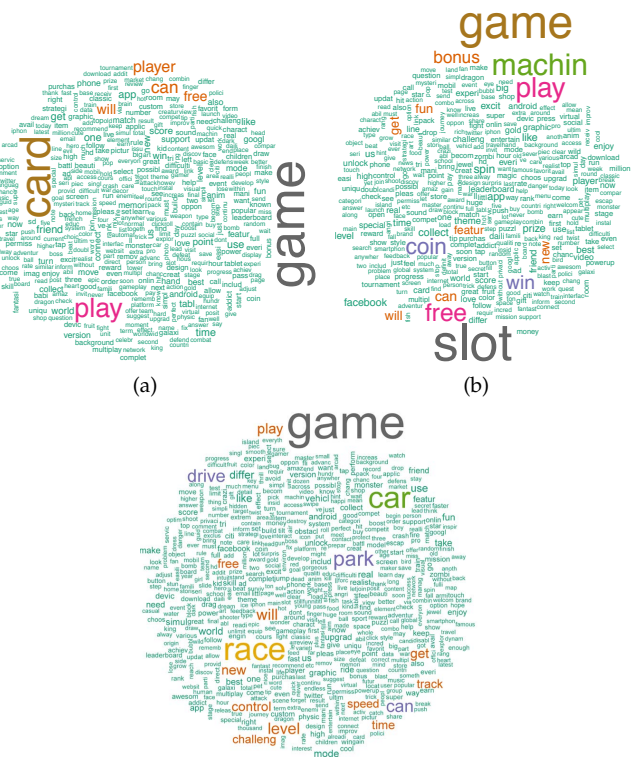


Fig. 11: (a) Word cloud for *Topic 4* - which corresponds to *Card* games, (b) Word cloud for *Topic 13* - corresponds to *Casino* games, (c) Word cloud for *Topic 8* - which corresponds to *Racing* games. The three figures suggest that the FRAC+'s categorization is semantically coherent.

to GOOGLE's categories.

We also evaluated the performance on categorizing the apps quantitatively. There are several performance metrics that can be used such as *perplexity* [33] to evaluate the first task. However, perplexity is more suitable for discrete probability mass function than continuous probability density function such as vMF distribution [56]. For this reason, we used normalized mutual information [36] (NMI),

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}}$$

where X represents cluster assignments, Y represents true labels on the data set, I and H represent mutual information and marginal entropy. NMI $\to 0$ represents bad clustering quality and NMI=1 represents perfect clustering quality. The NMI scores for FRAC+ and LDA are **0.284** and **0.217** respectively. FRAC+ returns a higher NMI score than LDA does. This result supports our qualitative results previously.

6.2 Detected Miscategorized Apps

We analyzed the miscategorized apps detected by FRAC+. As described in Section 4.3, we consider the miscategorized apps be the apps associated with the *smallest* vector component or from the top-P smallest vector components in the topic proportions θ_m in category m. Table 6 presents the number of miscategorized apps for scenarios where we take:

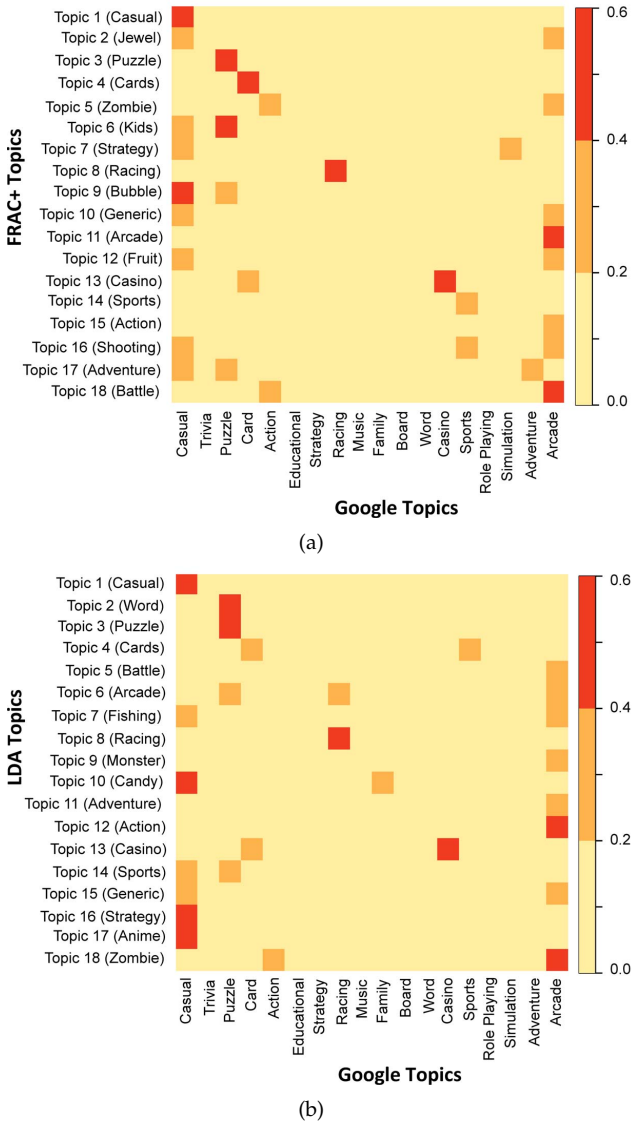


Fig. 12: The "heat map" between topics found from (a) FRAC+, (b) LDA, and new app categories from GOOGLE Play Store. The color intensity represents the percentage of game apps that belong to FRAC+'s topic and GOOGLE's category.

only from the (i) smallest (top-1), (ii) top-3 smallest and (iii) top-5 smallest vector components in the topic proportions θ_m of each category m. The results in Table 6 show that under the most conservative assumption (*i.e.* top-1) 0.35%–1.10% apps are found to be miscategorized in game app categories and under a more relaxed assumption 3.32%–11.08% apps are found to be miscategorized.

We visualize the percentage of the top-5 smallest vector components and the top-3 largest vector components based on the topics from FRAC+ in each GOOGLE's category in Figure 13. As can be seen, the topics taken from the smallest to top-5 smallest vector components are less related to the GOOGLE's category m. For instance, *Adventure* (0.3%), *Kids* (0.3%), *Zombie* (0.3%), *Puzzle* (0.3%) and *Casino* (0.6%) are less related to *Racing*.

Finally, we present some example miscategorized game

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

TABLE 6
Number of Detected Miscategorized Apps

Category	†Top-1	†Top-3	†Top-5
Arcade & Action	6 (0.35%)	44 (2.57%)	85 (4.96%)
Brain & Puzzle	5 (0.43%)	18 (1.53%)	39 (3.32%)
Cards & Casino	2 (0.54%)	12 (3.23%)	27 (7.28%)
Casual	7 (0.44%)	33 (2.08%)	76 (4.79%)
Racing	4 (1.10%)	15 (4.11%)	31 (8.49%)
Sports	3 (0.98%)	11 (3.29%)	37 (11.08%)

Note: †Top-P smallest vector components in the topic proportions θ_m of each category m. The percentage shows the proportion of miscategorized apps to the number of apps per category.

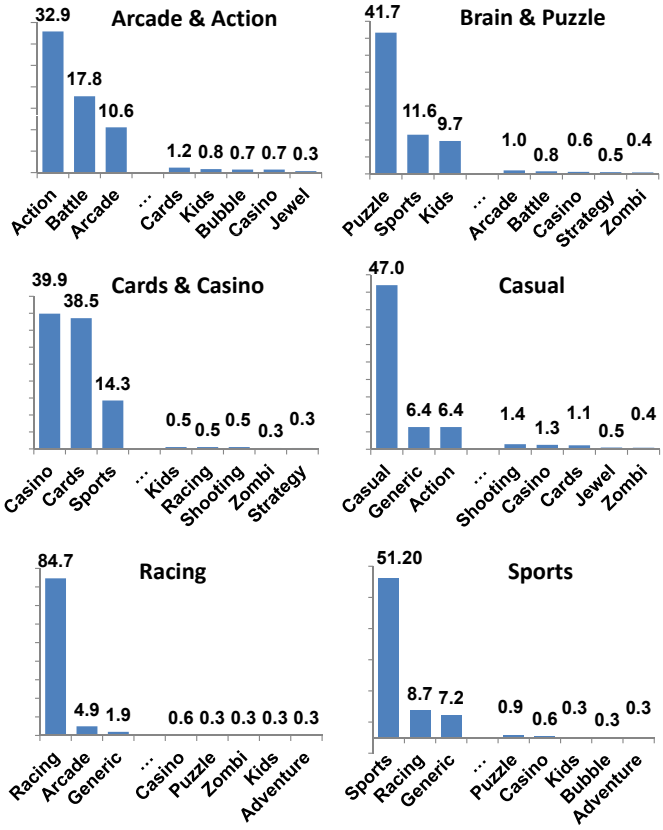
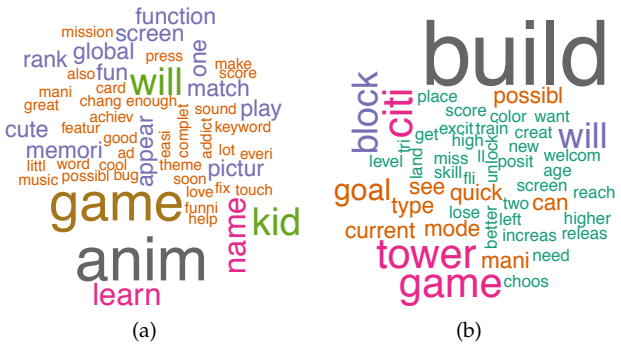


Fig. 13: Top-3 largest and top-5 smallest vector components based on FRAC+'s topics in the six GOOGLE's categories. All values are in percentages.

apps detected by FRAC+ in Table 7. Here we only include some apps from the *top-1* scenario described above. Due to space constraints, we present all apps in our supplementary material. In Figure 14 we visualize the word clouds of two miscategorized apps in *Racing* and *Cards & Casino* categories, to illustrate how the topics of these two apps are very different with the actual topics shown in Figures 11(a) and 11(b).

6.3 User Study

To obtain further insights on the accuracy of FRAC+, we carried out a small scale user study by recruiting paid



(a) (b)

Fig. 14: (a) Word cloud of *Alpha Zoo* (b) Word cloud of *Tower Blocks Classic*. *Alpha Zoo* was identified as miscategoriezed under *Racing* category and *Tower Blocks Classic* was identified as miscategorized under *Cards & Casino* category. (cf. Figure 11 for word clouds for *Racing* and *Cards & Casino*) categories.

users through the popular crowd-sourcing platform Crowdflower[12] and asking them to categorize a sample of 50 apps.

We only included users with high reliability scores given by the Crowdflower platform. To make sure that the users knew the details of the categories upfront and to avoid the users arbitrarily answering the survey, we took several precautionary actions. First, we included the definition of all categories in the beginning of survey. Then, before the survey was started, we trained the users by asking them to label five apps. These five apps were taken randomly from ten apps that we had chosen prior the survey to be correctly labeled. If the user failed to categorize them correctly, a feedback was given to explain why the answers were incorrect. The survey was started once the users were trained. To avoid any bias in their responses, we did not include the original category information of the apps in the questions.

The sample consisted of 26 apps that were identified as miscategorized by FRAC+ under *top-1* scenario described in Section 6.2. The sample also contained other 21 apps selected from the top list in the six game categories. We also included other three selected apps, which were correctly categorized from non-game categories. We included an app name, description and its screenshots for each question in the survey. Each user in the survey was then asked to select the appropriate category for the app from the categories *Arcade & Action*, *Brain & Puzzle*, *Cards & Casino*, *Casual*, *Racing*, *Sports Game*, and *Others*. Each app was evaluated by 20 users. For each app, we calculated the percentages of users who assigned the same category as the app's original category and users who did not.

In Figure 15, we show the users' responses for the 24 correctly categorized apps (21 apps from the top list and 3 apps from non-game category). As shown, over 80% of the users gave the same categories to 17 out of those 24 apps. Overall, over 50% of the users gave the same categories as the original apps' categories. These results suggest that in general the users agreed with the categories of the 24 popular apps.

[12]www.crowdflower.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2686851, IEEE Transactions on Knowledge and Data Engineering

TABLE 7
Some Detected Miscategorized Game Apps From GOOGLE Play Store Detected By FRAC+.

App Name	Original Category	App Description	Comment
Alpha Zoo	*Racing*	A for "Alligator" B for "Bear"; T for "Tiger" S for "Seal"! Wow!! There are so many lovely animals here! Little ones will have lots of fun with this addictive memory game learning letters and animals. How to play: To achieve the highest score match the pictures of the cute animals with their name's initial letter as soon as possible. When kids press one card in the grid a picture of animal or letter will appear responding with the sound of the animal makes or of the letter...	Based on the description this is a kids' alphabet game. Thus it is more appropriate to the category *Brain & Puzzle*.
Tower Blocks Classic	*Cards & Casino*	Welcome to Build The Tower! Exciting and colorful type of tower building games for all ages. A tower builder Tower Blocks will train your reflection. There are two modes to choose Build city & quick game: Build city: In Build City mode your goal is to create a thriving Megalopolis! Build towers and place them wisely in the city grid to reach the goal. Increasing your city's population and level will unlock new building types...	This game does not have any relevance to *Cards & Casino*. It is an *Arcade & Action* game which involves multiple levels.
Scratch Me	*Racing*	"Scratch Me" is the first and only virtual back scratch game for android. It allows the player to be in the virtual situation of helping someone be relieved by scratching their itchy back. In Scratch Me you have to find people in need of back scratching to help them get rid of their itch. When you help by scratching the in game characters you will be prompted what part of the back to scratch and will be told which ways and directions to change scratching....	This is a casual game and has no relevance to *Racing*
Plants vs Zombies Game Guide	*Casual*	Plants vs Zombies Game Guide Tips Cheats Free App This is my first app for Plants vs Zombies This tip will guide you and save your time & Money. This guide begins at the basic and change to difficulties methods and stratgies which will help you to cleared the level game. Tags: Plants vs Zombies cheats Plants vs Zombies help Plants vs Zombies videos ...	This app provides a guide to another game called Plants vs Zombies. Ideally this must be under non-game category.

Note that some game apps have been removed from GOOGLE Play Store and some the categories have been changed now.

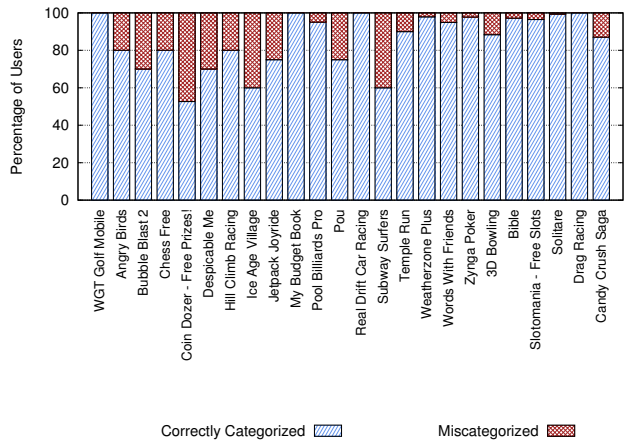


Fig. 15: User study results: Users' responses for the 24 correctly categorized apps. In majority, the users gave the same categories as the original apps' categories

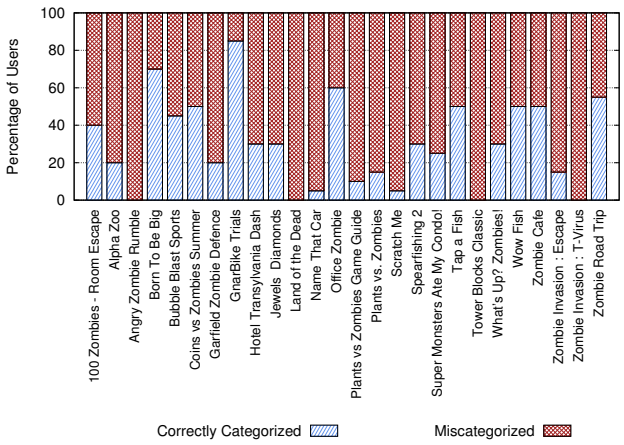


Fig. 16: User study results: Users' responses for the 26 miscategorized apps return by FRAC+. Over 50% of the users agreed that those apps were miscategorized.

Based on our experiments on GOOGLE Play Store data (Section 6.2), there are 26 apps[13], which were identified as miscategorized apps (we present the complete list in our supplementary material). Figure 16 shows that over 80% of the users gave different categories to 11 out of those 26 apps. In general, over 50% of the users gave different categories to 22 out of those 26 apps. These results suggest that we have only four false positives (∼85%) for the 26 apps identified as miscategorized by FRAC+, which support the performance of FRAC+ in identifying miscategorized apps.

[13]In Table 6 there are in total 27 apps under Scenario 1. However, among the results there are two instances of game app *Plants vs. Zombies*. Thus here we have 26.

7 CONCLUSION

We proposed FRAC+, (FR)amework for (A)pp (C)ategorization, to infer app categories and detect miscategorized apps. The key ideas include: (i) expressing the app descriptions as normalized word frequency counts which are modeled using a topic model based on directional distributions, (ii) integrating existing app categories with inferred categories. We have shown that our topic model is

able to effectively separate small number of apps that have different word distributions from the rest apps.

We performed extensive experiments on synthetic, semi-synthetic and real data sets to evaluate two main tasks: (i) detecting the correct number of categories, and (ii) detecting miscategorized apps. We created synthetic data with pre-defined number of topics and some noise. The experiment on synthetic data was designed to answer the first task. Our experiments show that FRAC+ detected the number of topics correctly comparing to the baseline methods (LDA and *k*-means++). We formed semi synthetic data by taking a subset of real data taken from GOOGLE Play Store and performed experiments to evaluate the second task. We shown that FRAC+ outperformed popular methods such as multi-class SVM, one-class SVM, LDA and *k*-means++ clustering algorithm in most settings, especially for larger size of data. Finally, we evaluated the performance of FRAC+ on real data extracted from GOOGLE Play Store. Several evaluation methods were used to validate FRAC+'s performance. FRAC+ gave higher normalized mutual information (NMI) score (0.284) than LDA did (0.217) on real data. We also showed that the categorization from FRAC+ is more aligned with GOOGLE 's new categories. Furthermore, the survey carried out using crowdsourcing showed that ∼85% of the users agreed with the miscategorization results returned by FRAC+.

Finally, FRAC+ model stems further research directions in related to topic modeling. In this work we considered a flat topic structure for apps as it is used in the current app markets. It is also interesting to extend the framework to handle hierarchical categories such as by using the nested Chinese Restaurant Process and to include the app source code analysis. We leave this research direction as our future work.

ACKNOWLEDGEMENTS

This work was supported by NICTA. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

REFERENCES

[1] C. Warren, "Google play hits 1 million apps," 2013, http://mashable.com.
[2] S. Fiegerman, "Apple's App Store tops 1 million apps," 2013, http://mashable.com.
[3] Unity3d, "5000+ apps banned by Apple," 2010, http://forum.unity3d.com.
[4] G. Machuret, "Aso ninja," 2013, primedia E-launch LLC.
[5] Barnes & Noble, "Nook app submission guide," 2013, https://nookdeveloper.zendesk.com.
[6] D. Sainati, "Evaluation and distribution of mHealth apps," in *HealthSummit*, 2014.
[7] Estoty Entertainment, "2048 Number puzzle game," 2014, https://play.google.com/store/apps/details?id=com.estoty.game2048.
[8] a3test, "2048," 2014, https://play.google.com/store/apps/details?id=com.a3test.tzfe.
[9] S. Perez, "Developer spams Google Play with ripoffs of well-known apps...again," 2014, https://techcrunch.com/2014/01/02/developer-spams-google-play-with-ripoffs-of-well-known-apps-again/.
[10] ——, "Nearly 60k low-quality apps booted from google play store in february, points to increased spam-fighting," 2013, https://techcrunch.com/2013/04/08/nearly-60k-low-quality-apps-booted-from-google-play-store-in-february-points-to-increased-spam-fighting/.
[11] S. Vakulenko, O. Müller, and J. v. Brocke, "Enriching iTunes app store categories via topic modeling," *The 35th International Conference on Information Systems*, 2014.
[12] A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM, 2016, p. 38.
[13] G. Berardi, A. Esuli, T. Fagni, and F. Sebastiani, "Multi-store metadata-based supervised mobile app classification," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC)*. ACM, 2015, pp. 585–588.
[14] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian, "Exploiting enriched contextual information for mobile app classification," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2012, pp. 1617–1621.
[15] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Mobile app classification with enriched contextual information," *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1550–1563, 2014.
[16] D. L. B. Lulu and T. Kuflik, "Wise mobile icons organization: Apps taxonomy classification using functionality mining to ease apps finding," *Mobile Information Systems*, 2015.
[17] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *International Conference on Software Engineering (ICSE)*, 2014, pp. 1025–1035.
[18] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: multi-layer profiling of Android applications," in *International Conference on Mobile Computing and Networking (MobiCom)*, 2012, pp. 137–148.
[19] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ""Andromaly": a behavioral malware detection framework for Android devices," *Journal of Intelligent Information Systems (JIIS)*, vol. 38, no. 1, pp. 161–190, 2012.
[20] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "Droidmat: Android malware detection through manifest and API calls tracing," in *Asia Joint Conference on Information Security (ASIAJCIS)*, 2012, pp. 62–69.
[21] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, "On the automatic categorisation of android applications," in *IEEE Consumer Communications & Networking Conference (CCNC)*, 2012, pp. 149–153.
[22] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun, "Active semi-supervised approach for checking app behavior against its description," in *IEEE 39th Annual International Computers, Software and Applications (COMPSAC)*, vol. 2. IEEE, 2015, pp. 179–184.
[23] A. Shabtai, Y. Fledel, and Y. Elovici, "Automated static code analysis for classifying Android applications using machine learning," in *Computational Intelligence Society (CIS)*. IEEE Computer Society, 2010, pp. 329–333.
[24] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Ranking fraud detection for mobile apps: A holistic view," in *Conference on Information and Knowledge Management (CIKM)*, 2013, pp. 619–628.
[25] R. Chandy and H. Gu, "Identifying spam in the iOS app store," in *Joint WICOW/AIRWeb Workshop on Web Quality*, 2012, pp. 56–59.
[26] 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 *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2013, pp. 1276–1284.
[27] M. Liu, C. Wu, X.-N. Zhao, C.-Y. Lin, and X.-L. Wang, "App relationship calculation: An iterative process," *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)*, vol. 27, no. 8, pp. 2049–2063, 2015.
[28] Y. Zhou, H. Yu, and X. Cai, "A novel k-means algorithm for clustering and outlier detection," in *IEEE International Conference on Future Information Technology and Management Engineering*, 2009, pp. 476 – 480.
[29] K.-A. Yoon, O.-S. Kwon, and D.-H. Bae, "An approach to outlier detection of software measurement data using the k-means clustering method," in *International Symposium on Empirical Software Engineering and Measurement*, 2007.
[30] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in

Advances in Neural Information Processing Systems (NIPS), 2000, pp. 582 – 588.

[31] L. M. Manevitz and M. Yousef, "One-Class SVMs for document classification," *Journal of Machine Learning Research (JMLR)*, vol. 2, pp. 139 – 154, 2001.

[32] V. Barnett and T. Lewis, *Outliers in statistical data*. John Wiley & Sons Ltd., 1978.

[33] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty, "Latent Dirichlet Allocation," *Journal of Machine Learning Research (JMLR)*, vol. 3, pp. 993–1022, 2003.

[34] A. Ahmed, Y. Low, M. Aly, V. Josifovski, and A. J. Smola, "Scalable distributed inference of dynamic user interests for behavioral targeting," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2011.

[35] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora," in *Conference on Empirical Methods in Natural Language Processing*, 2009, pp. 248 – 256.

[36] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von Mises-Fisher distributions," *Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1345–1382, 2005.

[37] D. Surian and S. Chawla, "Mining outlier participants: Insights using directional distributions in latent models," in *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) (3)*, 2013, pp. 337–352.

[38] C. C. Aggarwal, S. C. Gates, and P. S. Yu, "On the merits of building categorization systems by supervised clustering," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 1999, pp. 352–356.

[39] X. Cao, G. Cong, B. Cui, C. S. Jensen, and Q. Yuan, "Approaches to exploring category information for question retrieval in community question-answer archives," *ACM Transactions on Information Systems (TOIS)*, vol. 30, pp. 7.1–7.38, 2012.

[40] Q. Yuan, G. Cong, A. Sun, C.-Y. Lin, and N. M. Thalmann, "Category hierarchy maintenance: a data-driven approach," in *ACM SIGIR International Conference on Research and Development in Information Retrieval*, 2012, pp. 791–800.

[41] L. He and X. Sun, "Automatic maintenance of the category hierarchy," in *International Conference on Semantics, Knowledge and Grids (SKG)*, 2013, pp. 218–221.

[42] B. Li, J. Liu, C.-Y. Lin, I. King, and M. R. Lyu, "A hierarchical entity-based approach to structuralize user generated content in social media: A case of Yahoo! answers," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1521–1532.

[43] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. Mcgraw-Hill College, 1996.

[44] K. V. Mardia and P. E. Jupp, *Directional Statistics*. John Wiley and Sons, Ltd., 2000.

[45] J. Yuan, Y. Zheng, and X. Xie, "Discovering regions of different functions in a city using human mobility and POIs," in *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2012, pp. 186–194.

[46] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics (JCAM)*, vol. 20, no. 1, pp. 53–65, 1987.

[47] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.

[48] D. Ramage and E. Rosen, "Stanford topic modeling toolbox," 2009, http://nlp.stanford.edu/software/tmt/0.4/.

[49] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[50] S. Seneviratne, A. Seneviratne, M. A. Kaafar, A. Mahanti, and P. Mohapatra, "Early detection of spam mobile apps," in *International Conference on World Wide Web Conference (WWW)*, 2015.

[51] Google Inc., "FAQ for upcoming change to game categories," 2014, https://support.google.com.

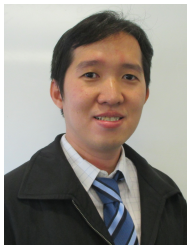
[52] Statista, Inc., "Most popular Google Play App Store categories in 4th quarter 2012, by share of listed apps," 2012, http://www.statista.com.

[53] ——, "Most popular Apple App Store categories in March 2014, by share of available apps," 2014, http://www.statista.com.

[54] M. H. DeGroot, *Probability and statistics (2nd Ed)*. Addison-Wesley, 1986.

[55] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *ACM Special Interest Group on Management of Data (SIGMOD)*, 2012.

[56] C. Elkan, "Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution," in *International Conference on Machine Learning (ICML)*, 2006, pp. 289–296.



Didi Surian Didi Surian is a postdoctoral research fellow at the Australian Institute of Health Innovation, Macquarie University. He was with the School of Information Technologies, the University of Sydney and NICTA. His research interests are data mining, machine learning, outlier detection and social network analysis.

Suranga Seneviratne Suranga Seneviratne is a Researcher at the Networks Research Group at Data61, CSIRO. He received his PhD from the University of New South Wales and his research interests are in privacy and security in mobile systems, applied machine learning and data mining to solve networking problems.

Aruna Seneviratne Aruna Seneviratne is the foundation Chair in Telecommunications and holds the Mahanakorn Chair of Telecommunications in the School of Electrical Engineering and Telecommunication at the University of New South Wales. He is also the Research Director of the Cyber Physical Systems Research Program at Data61, CSIRO. His current research interests are in mobile content distributions and preservation of privacy. He received his PhD in electrical engineering from the University of Bath, UK. He has held academic appointments at the University of Bradford, UK, Curtin University, and UTS.

Sanjay Chawla Sanjay Chawla is with Qatar Computing Research Institute (QCRI), HBKU, Doha, Qatar. Prior to joining QCRI, Sanjay was a Professor in the School of Information Technologies, the University of Sydney, Australia. His main area of research is data mining and machine learning. More specifically he has been working on three problems of contemporary interest: outlier detection, imbalanced classification and adversarial learning. His research has been published in leading conferences and journals and has been recognized by several best-paper awards. He serves on the editorial board of IEEE TKDE and Data Mining and Knowledge Discovery. Sanjay served as the Head of School from 2008-2011 and was an academic visitor at Yahoo! Labs, Bangalore in 2012. He served as PC Co-Chair for PAKDD 2012 and SDM 2016.