國立政治大學資訊管理學系

碩士學位論文

指導教授:郁方博士

AppReco: 基於行為識別的行動應用服務推薦系統

AppReco: Behavior-aware Recommendation for iOS Mobile Applications

研究生：方子睿

中華民國一零五年六月

摘要

在現在的社會裡，手機應用程式已經被人們接受與廣泛地利用，然而目前市面上的手機 App 推薦系統，多以使用者實際使用與回報作為參考，若有惡意行為軟體，在使用者介面後竊取使用者資料，這些推薦系統是難以查知其行為的，因此我們提出了 AppReco，一套可以系統化的推薦 iOS App 的推薦系統，而且不需要使用者去實際操作、執行 App。

整個分析流程包括三個步驟：(1) 透過無監督式學習法的隱含狄利克雷分布(Latent Dirichlet Allocation, LDA)做出主題模型，再使用增長層級式自我組織映射圖(Growing Hierarchical Self-Organizing Map, GHSOM)進行分群。(2)使用靜態分析程式碼，去找出其應用程式所執行的行為。(3)透過我們的評分公式對於這些 App，進行評分。

在分群 App 方面，AppReco 使用這些應用程式的官方敘述來進行分群，讓擁有類似屬性的手機應用程式群聚在一起；在檢視 App 方面，AppReco 透過靜態分析這些 App 的程式碼，來計算其使用行為的多寡；在推薦 App 方面，AppReco 分析類似屬性的 App 與其執行的行為，最後推薦使用者使用較少敏感行為(如使用廣告、使用個人資料、使用社群軟體開發包等)的 App。

而本研究使用在 Apple App Store 上面數千個在各個類別中的前兩百名 App 做為我們的實驗資料集來進行實驗。

關鍵詞：推薦系統、手機應用程式、主題模型

ABSTRACT

Mobile applications have been widely used in life and become dominant software applications nowadays. However there are lack of systematic recommendation systems that can be leveraged in advance without users' evaluations. We present AppReco, a systematic recommendation system of iOS mobile applications that can evaluate mobile applications without executions.

AppReco evaluates apps that have similar interests with static binary analysis, revealing their behaviors according to the embedded functions in the executable. The analysis consists of three stages: (1) unsupervised learning on app descriptions with Latent Dirichlet Allocation for topic discovery and Growing Hierarchical Self-organizing Maps for hierarchical clustering, (2) static binary analysis on executables to discover embedded system calls and (3) ranking common-topic applications from their matched behavior patterns.

To find apps that have similar interests, AppReco discovers (unsupervised) topics in official descriptions and clusters apps that have common topics as similar-interest apps. To evaluate apps, AppReco adopts static binary analysis on their executables to count invoked system calls and reveal embedded functions. To recommend apps, AppReco analyzes similar-interest apps with their behaviors of executables, and recommend apps that have less sensitive behaviors such as commercial advertisements, privacy information access, and internet connections, to users.

We report our analysis against thousands of iOS apps in the Apple app store including most of the listed top 200 applications in each category.

Keywords: Recommender System, Mobile Application, Topic Model

Contents

List of Figures

List of Tables

1 Introduction

Mobile applications on iOS and Android platforms have been increasing remarkably in the past decade and have become the dominant software applications in history. According to the reports of Statistics [2, 3], both Google Play and Apple's App Store have over 1.5 million apps available for public download. In the recent annual Apple conference, WWDC 2015, Tim Cook announced that the number of app downloads in Apple's App Store has crossed 100 billions.

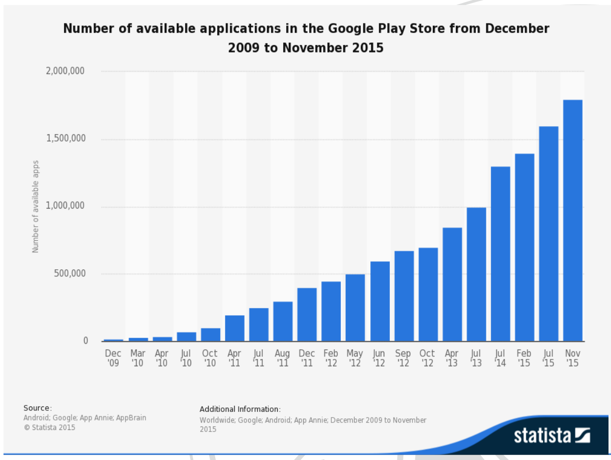


Figure 1: Number of Available Apps in the Google Play Store

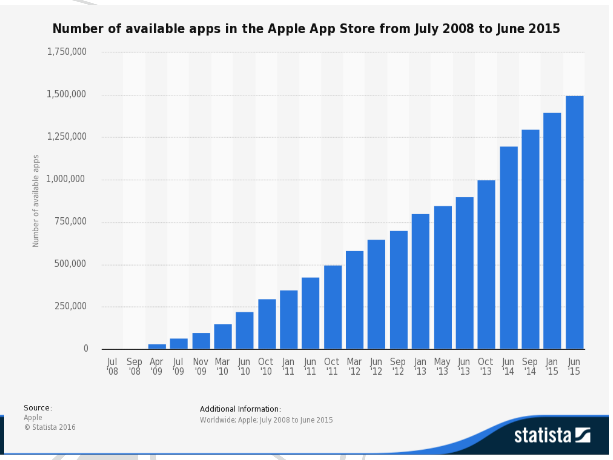


Figure 2: Number of Available Apps in the Apple App Store

To help users find suitable mobile applications, various recommendation systems have shown up and most likely the important one, e.g., Editor's choice of Apple's App store, could dominate the trend of mobile applications. Most common recommendation systems are based on user experiences, using writers reviews or users' ratings to judge apps' performance [4, 5, 6]. These systems can be classified as three types according to how they work. The first group is content-based filtering recommendation systems, which recommend apps based on how app descriptions, features and contents match the preference of target users. The second group is collaborative filtering recommendation systems, which recommend apps by focusing on a target users interests similarity with other users. The last group is hybrid filtering recommendation systems, which mix the concepts of content-based and collaborative filtering technique, recommending apps from the both different aspects.

Recommendation systems that are based users' reviews are limited to users' observations. It requires users to use the app before evaluating the apps. It is hence rather hard to scale out to handle a numerous number of apps that appear together in a limited time. That is to say, less popular apps would lack of reviews to be evaluated. It is also hard to have the same standard from different views of article writers or users without systematic recommendation mechanisms. Furthermore, users may not be able to observe all the behaviors of apps, particularly for those behaviors that may not show up in the user interface or are only triggered under a specific circumstance. A famous example is a very popular iOS application, called PATH, which is used to record location paths of users in daily life. Path has been reported to upload user's address book to its remote server without authorization from users [7]. Note that PATH is popular and widely recommended before its malicious behavior is revealed.

In fact, the number of malicious software on mobile platforms has kept increasing in recent years. A recent mobile software analysis report which shows in Fig. 3 from Germany [8] shows that from January 2015 to September 2015, the reported malware has crossed the number of the whole 2014 year. The report from McAfee [9] also pointed out a remarkably increasing rate in growth of mobile malware.

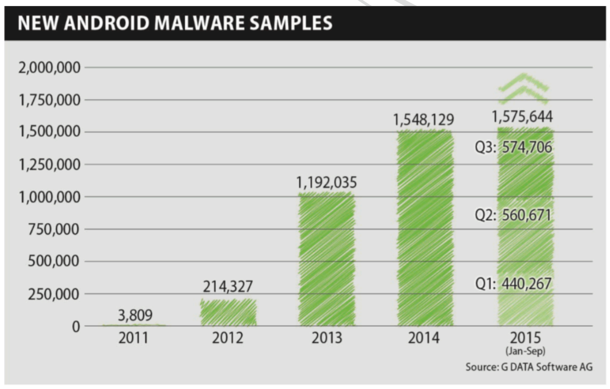


Figure 3: New Android Malware Sample from G-Data

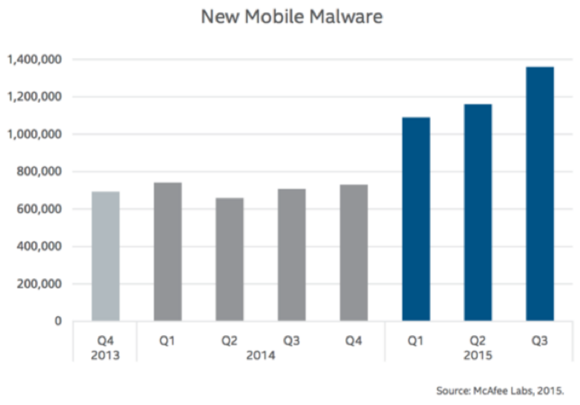


Figure 4: New Mobile Malware from McAfee

We propose in this work a systematic recommendation system that takes app behaviors into account. We summarize four behaviors of apps that may raise serious concerns of users or app publishers: (1) taking user's privacy data,(2) using third party SDK (Software Development Kits), (3) prompting advertising, and (4) using private APIs (Application Programming Interfaces). The privacy information of a user such as address book, calendar, photo, has been considered to be one of the most important concerns for mobile users [10, 11]. Mobile applications equipped with the third party SDKs may be used to access user data from the third party web services, e.g., Facebook and Google plus, both offer iOS SDKs to developers to develop functions to fetch data. While mobile ads play an important role in creating revenue for developers, e.g., in 2013, mobile revenue has reached over 17 billion [12], and analysts reported that revenue from mobile advertisements would have surpassed television advertising in 2017 [13], those ads may also cause negative effects to users [14]. Finally, private APIs are designed for internal usage of Apple developers. To vet applications for the usage of private API, Apple creates a system, called App Review, to detect iOS applications for their usage of private APIs. However, recent researches have indicated that there are still some Apps using private APIs in the Apple Store [15].

The above behaviors may not show out in app descriptions or user interface. They could be triggered under certain conditions and could not be observed by users. Recommendation systems that depend on user experiences are not be able to take these behaviors into account without being aware of these behaviors. We adopt static binary analysis to reveal potential behaviors that are embedded in the executable. First, we extract out executable code that includes class, method names from each application with IDA Pro. These function calls are primary from Objective-C, several from C. Second, we construct the behaviors and private API list from Apple developer library and iOS runtime headers [16, 17]. Third, we compare the executable code from application and pattern list. We could detect whether applications perform these behaviors or not.

We have realized the idea in AppReco, a novel behavior-aware recommendation service for iOS mobile applications. AppReco ranks similar interest apps via risk evaluation according to their behavior and the weighted formula that is designed under the idea of term frequency and inverse document frequency. To find similar interest apps, AppReco adopts unsupervised topic discovery and clustering to cluster similar applications. This is done by systematic downloading application information of each application from Apple App Store, including name, description, genre and ratings. The descriptions of applications are then pre-processed to terms-document matrix and then be used to extract topic models via Latent Dirichlet Analysis. The extracted models identify the topics of app descriptions, as well as the topic (in the format of a percentage distribution among topics) of each app. We then apply the unsupervised clustering (the GHSOM algorithm) on apps according to their topic distributions. Apps that fall in the same cluster have similar topic distribution in their descriptions and are considered to have the same interests.

We have evaluated AppReco against thousands of apps that are directly downloaded from Apple App Store. These applications are automatically downloaded, decrypted and analyzed, where most of which were from the top download list. The preliminary results show that without expertise and user interventions, we are able to recommend users mobile applications that have similar topics with less undesired behaviors.

2 Literature Review

For the purpose of building behavior-aware recommendation system, we have to find an access to generate ideal recommendation results. In related work, at first, in order to construct a mobile recommendation system, we investigate mobile recommendation system and common recommendation systems techniques. Second, we study some topics model techniques of natural language processing to convert applications' descriptions to vectors so that we could cluster applications with vectors. Finally, we study what is the sensitive behaviors of mobile application and how to find out behaviors of mobile applications.

2.1 Mobile Recommendation System

Nowadays, due to a huge number of mobile applications, recommendation system for mobile applications is influential because users look for apps that meet the requirements through it. Mobile application recommendation systems have been increasing not only at the industry but also at the research community [18, 6, 4, 5].

AppBrain [18] is an application discovery tool on Android platform. It monitors the installation history of applications of users and provides the recommendation in the same category. It also provides all-time popular applications to users.

The Sweet Setup [6] provides mobile application discovery service for iOS users. Their recommendations are sorted within app categories.

Apps4Review [4] is a community, where developers submit their iOS or Android applications to the authors of a website. The authors will write the application reviews and articles.

AppJoy [5] makes personalized application recommendations on Android platform. Its function is based on measuring how users operates their installed applications. With the usage data, it practices item-based collaborative filtering recommendation systems.

After surveying recommendations of mobile applications, we find out there are little amount of systems focusing on behaviors of applications. For example, authors of recommendation systems review applications manually, but it is hard to tackle a large scale data. Therefore, we develop a novel recommendation system, that automatically takes mobile behavior into consideration for recommendation.

2.2 Recommendation System

Recommendation system is a platform that provides users with suggestion of items. With recommendation systems, users save their times on searching similar items and approaching items ratings and comments. In recent years, recommendation system technique has been widely applied to movies, music, games, books, news, products and so on [19, 20, 21]. Suggestions from recommendation systems may be based on users preference for items, similarity within items, rating and etc.

Balabanović and Shoham [22] classified recommendation systems according to how they recommend items. They divided these recommendation systems into three categories: content-based recommendation system, collaborative filtering recommendation system and hybrid recommendation system. The first type is content-based recommendation system which recommends items similar to those a given user prefers. The another type is collaborative filtering recommendation system which identifies users whose tastes are similar to those of given users and recommends items they like. The other type is hybrid recommendation system which combined the content-based with collaborative filtering recommendation system to incorporate the advantages while inheriting the disadvantages of neither side. In the following section, we investigate these three recommendation systems.

2.2.1 Content-based Recommendation

Content-based recommendation systems suggest items based on a users preference, interest and description of an item. For example, this kind of systems would recommend an item if the item is similar to some items that a user has preferred before.

The content-based approach roots in information retrieval and information filtering research [23]. First, with technique in information retrieval and information filtering, system could transfer item attribute, description, and content to a set of feature vectors from item. Then, it computes cosine value between two items with their vector to measure the similarity. If their cosine similarity is close to 1, it means that they are similar to each other. In contrast, if their cosine similarity is far away from 1, it means that they are totally different.

Content-based system uses text mining techniques and focuses on text-based item, such as news, websites, documents, and etc. The most used text mining approach is Term Frequency and Inverse Document Frequency (TF-IDF), it uses term frequency in document to export the characteristic of document. After the information retrieval, content-based recommendation systems also consider the information about users tastes, preferences, and needs.

There are some popular recommendation systems adopting content-based approaches such as Rotten Tomatoes a movie recommendation platform and Pandora Radio a personalized music platform. Some research and study apply content-based technique to their systems.

Pazzani, Muramatsu and Billsus [20] developed a content-base recommendation software, Syskill & Webert, recommending websites which might interest a user. First, they created a user profile which contained user's preference by training data such as interested pages and uninterested pages. Second, they indexed each website with topics, so they could understand user's latent topic interest. Finally, they could recommend user a website based on the topic preference.

Van Meteren and Van Someren [21] also built up a websites recommendation systems, called PRES by using cosine measurement between user profiles and documents vector. They constructed user profiles by a learning algorithm, which considered the users consumed time on reading a website.

2.2.2 Collaborative Filtering Recommendation

Collaborative filtering recommendation systems suggest items to a user based on opinions from other users. For example, systems would recommend an item to a user because another user with similar tastes and preferences has picked this item before. Generally, collaborative filtering is to predict the taste of a particular user from a database of user votes from a sample or population of other users. According to different predict approach, Breese, Heckerman and Kadie [24] have grouped collaborative filtering recommendation system into two types, 1. Memory-based collaborative filtering 2. Model-based collaborative filtering.

In memory-based collaborative filtering algorithms, they predict the preference of a user based on partial information regarding the user. By applying the similarity between users or items, memory-based collaborative filtering can be separated into user-based type and item-based type.

User-based collaborative filtering is based on users preference similarity. If some users have similar preference on some items, system would consider that the users would also share similarity on other items preference.

For example, Resnick, Iacovou, Suchak, Bergstrom and Riedl [19] developed GroupLens which was a long history of user-based collaborative filtering and had lots of recommendation system projects. In the original GroupLens, they used the technique to make a news article platform to help people find news and articles in which they are interested. At first, they asked users to assign their numeric ratings after reading news or articles. These ratings would be used in two ways. First, they computed the ratings to determine the preferences of users to find out which users ratings were most similar to each other. Second, they would predict what kinds of users would be interested in news or articles, based on ratings from similar users. Finally, they could suggest the user news and articles which met the users interest.

In order to determine the preferences of users, if the number of users has increased, the computing time of user-based recommendation system would be comparatively long. Sarwar, Karypis, Konstan, and Riedl [25] proposed item-based collaborative filtering which was based on item feature similarity such as rating history. First, they computed the similarity between different items. Then, they used the data of user rating on different items to predict the user's preference for the specific item. For example, if there were high-rated histories of item A and item B, the system would recommend users item A when users were viewing item B and vice versa.

Memory-based collaborating filtering approaches use many storage space to store customer preference data, but it is hard to deal big data in real-time. To conquer the above weakness, there is the other type of memory-based collaborating filtering, model-based collaborating filtering [24]. Model-based collaborative filtering is based on using some data mining, machine learning algorithms to train data, and then using results to make recommendation. In contrast to memory-based collaborating filtering, model-based approach calculates the expected value of a vote with training model. With proper models, it can discover some latent factor behind observed users ratings, and make a good recommendation. There are many algorithms able to perform model-based collaborative filtering, such as Latent Semantic Analysis [26], Probabilistic Latent Semantic Analysis [27], Latent Dirichlet Allocation [1] and so on.

Krestel, Fankhauser and Nejdl [28] developed a tag recommendation system by using Latent Dirichlet Allocation and proved their performance better than using traditional association rules. First, they used LDA to elicit latent topics from resources and then made a set of tags to recommend topics for new resources with only a few tags. Second, they sorted out the tags belonging to topics so that they could recommend tags to a specific topic. Finally, the recommended tags are specified for a particular resource, and thus useful for searching and recommending resources to other users.

Hofmann [29] used probabilistic latent semantic analysis to apply to movie data and convinced that model-based collaborative filtering using pLSA would perform better than memory-based technique. He used the probabilistic latent semantic models to compress the data into a compact model that could automatically identify user communities and interest groups and then compute preference prediction in constant time.

When dealing with large data sets, model based collaborative can show up higher performance and lower processing time for a recommendation.

2.2.3 Hybrid Recommendation System

Both content-based and collaborative filtering recommendation systems have their advantages and disadvantages. To combine the advantage of the above two approaches, hybrid recommendation system was invented.

Adomavicius and Tuzhilin [23] grouped hybrid recommendation system into four types:

1. Implement collaborative and content-based methods separately, using linear function to combine result and deciding final recommendation result.

2. Incorporate some content-based characteristics into a collaborative approach, using the users' before preference as data to calculate the similarity within users.

3. Incorporate some collaborative characteristics into a content-based approach, using some dimensionality reduction technique to content-based profile to create a collaborative view of a collection of user profiles.

4. Construct a general unifying model that incorporates both content-based and collaborative characteristics, e.g. adopting probabilistic model or Bayesian mixed-effect regression models to combine two approaches.

Some researchers try to combine content-based and collaborative filtering to gain performance in some cases [30, 31]. De Campos, Fernández-Luna, Huete, and Rueda-Morales [31] presented a new Bayesian network model to deal with the problem of hybrid recommendation system by combining content-based and collaborative features. They used Bayesian network to construct the relationships between users, items, features, and then used the strength of these relationships to recommend movies. First, they sorted out the active user votes the items to construct the user profile, which would include the information that content-based and collaborative approaches needed. Then they built up relationships related to the target item with user profiles. In the end, they could make predictions and recommendations to active users with relation models.

Yoshii, Goto, Komatani, Ogata, and Okuno [30] developed a hybrid recommendation system for music by combining content-based and collaborative filtering technique. They used probabilistic models to unify content-based and collaborative filtering approach to get their advantage. With content-based technique, the system could deal with music without rating situation which might make collaborative filtering perform badly. With collaborative filtering approach, the system could consider users' preference in terms of music contents which were hard to be fully investigated.

2.2.4 Comparison of Recommendation Systems

After listing the above methods of recommendation system, we have to choose the most suitable approach on our research. We find a research in 2005 by Adomavicius and Tuzhilin [23], it compared the three recommendation methods, and pointed out their advantage and disadvantage respectively. The main difference between content-based and collaborative filtering system is that the former approach focuses on measuring the similarity between vectors of items, while the latter targets on calculating the similarity between vectors of the actual user-specified ratings. In AppReco, we focus on how the similarity within the applications, so we decide to choose content-based approach as our main method.

After surveying some content-based recommendation systems, we find out that some of them use topics model to make a recommendation model. For example, Godin, Slavkovikj, De Neve, Schrauwen and Van de Walle [32] used the Latent Dirichlet Allocation (LDA) to model the underlying topic assignment of language classified tweets. They used over 1.8 million tweets as data set, and randomly picked 100 tweets of them for constructing model. They ended up proving that they could recommend hashtags for tweets in a fully unsupervised manor. In our research, we are going to find out topic model approaches to construct our recommendation model.

2.3 Topic Model

AppReco uses the topic model technique in natural language processing (NLP) to convert the applications' descriptions into a vector of word appearances. The result of topic model can reflect a collaborative shared view of the application and popular vocabulary of the topics to describe applications. In addition, topics model explains the feature vector of data so that we can perform clustering approach. In this part, we investigate three popular topic model methods. They are used to dig out latent keywords in a document.

2.3.1 Latent Semantic Analysis

Deerwester, Dumais, Furnas, Landauer and Harshman [26] proposed Latent Semantic Analysis (LSA), a famous information retrieval technique in 1990. The purpose of LSA, also called Latent Semantic Indexing(LSI) is analyzing documents to find the latent meaning or concepts behind the documents. LSA assumes that terms will have high-related meaning in similar documents and that terms will have high association with concepts and meanings.

However, the problem is difficult, because a word often bears multiple meanings. Lexical ambiguity obscures concepts of a single term. Even native speakers would have a hard time dealing with the received information. For example, the term "case" means a criminal institution when it was used around "murder", "police", and "victim". In contrast, when it was used around "package", "luggage", and "travel", it probably indicates a suit case.

In order solve the ambiguity, LSA maps both terms and documents into a matrix space and does the comparison in this space at first. Then, LSA uses a mathematical technique, called Singular Value Decomposition (SVD) model to reduce the number of rows while preserving the feature of structure with the large matrix. The core formula of SVD is Eq.(1).

$$X = T_0 S_0 D_0^{'} \tag{1}$$

In Eq.(1), X represents a $t \times d$ matrix of terms and documents, decomposed into the product of three matrix. T_0 and $D_0^{'}$ have orthonormal columns and S_0 is diagonal. In SVD, T_0 and $D_0^{'}$ are the matrices of left and right singular value vectors and S_0 is the matrix of diagonal matrix of singular value. At same time, the singular values of S_0 are ordered by sized. They mean the eigenvalues of the terms and documents. After SVD decomposing the matrix, LSA uses the number of the dimension k to reduce dimensional representation of the matrix. It can emphasize the strongest relationships and throw away the noise. Thus, the Eq.(1) would be change to Eq.(2).

$$X \approx \widehat{X} = T_k S_k D_k^{'} \tag{2}$$

With Eq.(2), it reserves large k entries in Eq.(1) and the product of the result matrices, a matrix $\widehat{X}$ which is approximate to X. The number of dimensions k is very important. If k is too small, it would cause important pattern to be dropped out. In contrast, if k is too big, it would let noise word creep in. In the end of SVD, T_k would become the term-topics matrix while $D_k^{'}$ would become the topics-documents matrix.

Landauer, Foltz and Laham [33] applied LSA to knowledge with Grolier Encyclopedia, a popular academic encyclopedia in the United States. They used 60,768 terms and 30,473 documents from Grolier to construct a terms - documents matrix. Then, they applied SVD to reduce the high-dimension matrix and to reconstruct with a low-dimensional matrix. Finally, they could compute the latent concept behind terms with the low-dimensional matrix. They concluded that LSA would perform well around 300 dimensions, and perform comparatively poorly below 100 dimensions so that dimensionality-optimization could greatly improve the extraction and representation of LSA.

2.3.2 Probabilistic Latent Semantic Analysis

Hofmann [27] proposed a Probabilistic Latent Semantic Analysis (pLSA), also known as Probabilistic Latent Semantic Indexing (pLSI) in 1999. In contrast to LSA, a document is represented as a mixture of the topics in pLSA. The pLSA uses the aspect model to show a low-dimensional representation of the co-occurrence data, associated with hidden variables. The creation of pLSA model differs from LSA, which usually uses a singular value decomposition to reduce down a high dimension matrix. pLSA is based on a mixture decomposition derived from a statistical technique, latent class model. The model of pLSA is Fig. 5.

d and w represent observable documents and words index variable. z represents a latent variable which is the topic distribution of documents and words. M means the

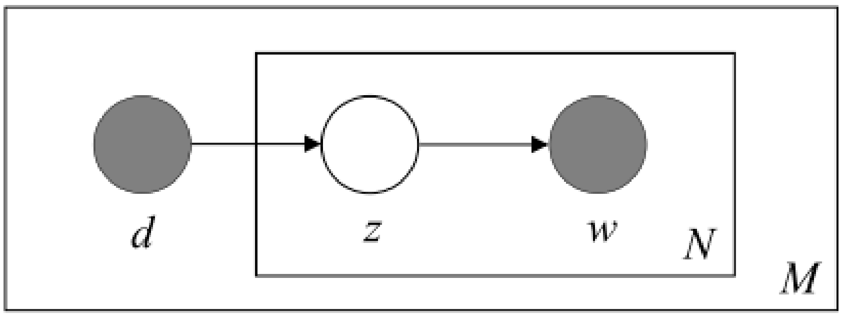


Figure 5: Model of pLSA. [1]

number of the documents and N means the number of words of one of documents d.

The process of pLSA can be divided into 3 steps and represented as an Eq.(3):

$$\begin{aligned} P(d,w) &= P(d)P(w|d) \\ P(w|d) &= \sum_{z} P(w|z)P(z|d) \end{aligned} \tag{3}$$

1. Select a document d with probability $P(d)$.
2. For each words in document d, select a topic z with probability $P(z|d)$.
3. Select word w on the previous chosen topic z with probability $P(w|z)$.

In order to estimate the latent variable in pLSA, Hofmann [27] uses Expectation Maximization (EM) algorithm. EM can be divided into three steps: 1. computing posterior probabilities for the latent variables z with the current estimates of the parameter 2. using the parameter computed in the previous step to maximize estimation parameter, and 3. repeat step 1 step 2 until convergence.

In the conclusion, Hoffman proved that pLSA could take advantage of statistical standard methods for model better than traditional LSA. pLSA could be applied to language modeling, collaborative filtering.

2.3.3 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA), a type of topic model was proposed by Blei, Ng and Jordan [1] in 2003. The general idea of LDA is based on a hypothesis that a document may be viewed as a mixture of various topics. LDA constructs topic model with probabilistic statistic model. LDA is very similar to pLSA, but LDA assumes two Dirichlet priors to the topic distribution. In practice, LDA reduces more executing time than pLSA because pLSA has to estimate lots of parameters. If a user gives a uniform Dirichlet prior to the LDA model, it would be equivalent to the pLSA model. The model of LDA is Fig. 6.

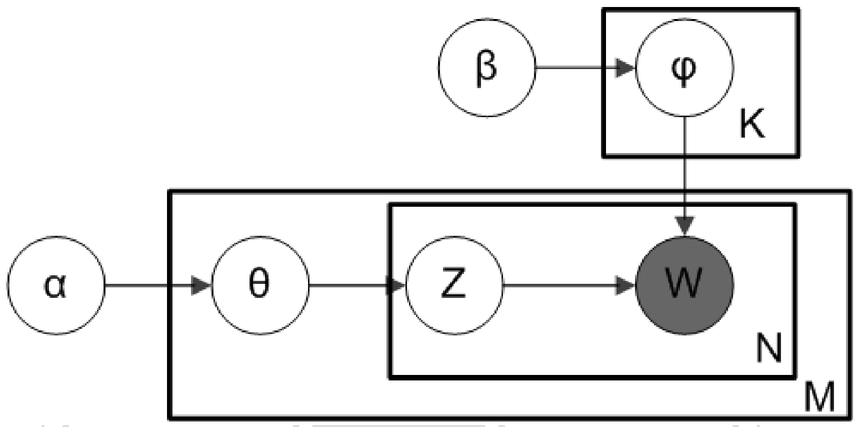


Figure 6: Model of LDA. [1]

In the Fig. 6, parameter α is the Dirichlet prior of topic distributions of documents. β is the Dirichlet prior of word distributions of topics. K is the number of topics. M is the number of documents. N is the number of words in a document i. θ is a $M \times K$ matrix which means topic distributions for each document. φ is a $K \times W$ matrix which means word distributions for each topic. z is a set of topics of the document i, and w is the only observable variable, a set of word of topics in the document i.

The following is the generative process of LDA, which would be applied to each document in corpus:

1. Choose N, the number of words in a document i.

2. Choose the topic distribution θ of the the document, with the Dirichlet prior α.

3. Choose the word distribution φ of the the topic, with the Dirichlet prior β.

4. For each word j in the document i, choose a topic $z_{i,j}$ with multinomial distribution θ_i.

5. For each word j in the document i, choose a word $w_{i,j}$ with multinomial distribution $\varphi_{z_{i,j}}$.

After repeating the generative process of LDA, it would reach a convergence and steady result model. As a result, they proved that LDA model had a better prediction result than the mixture of unigrams model and the pLSA model.

2.3.4 Comparison of Topic Models

After the instigation of approaches of topic model, it is obvious that each approach has its unique strength and inevitable weakness. In order to find the most qualified method for our research, we filter out the less suitable methods with the comparison step by step. Each points of comparison would be discussed separately in the following paragraphs.

For statistics, LDA and pLSA have strong enough background to graph their model in Fig. 5 and Fig. 6. They both use the concept of probability and their advantage is that the topics, extracted by these models are invariably interpretable. LDA and pLSA facilitate the analysis of model output, unlike the uninterpretable directions produced by LSA.

For efficiency, LSA has the lowest executing time than LDA and pLSA because it doesn't need to calculate the probability of model. LDA is the second efficient method because it estimates the probability fast with its hyper-parameter Dirichlet priors to the topic distribution. On the contrary, pLSA has to estimate all probability in their equation so it has the lowest efficiency among all the methods.

After comparing, we first filter out LSA because its lack of statistical model. Second, we filter out pLSA because LDA has better efficiency on executing. As a result, we use LDA as our topic model method.

2.4 Behavior Analysis

2.4.1 Sensitive Behaviors of Mobile Applications

Nowadays, there are lots of mobile applications in the markets for satisfying users need. Besides, an application often bears multiple functionalities, called mobile behavior of the mobile application. Mobile application developers design these mobile behaviors to cater to users and gain profit recently. However, if a developer placed a malicious behavior into an application or misused the user information from an application, user's privacy information would be leaked or be recorded to database which is handled by hackers or malicious developers.

Felt, Finifter, Chin, Hanna, and Wagner [34] classified mobile threats model into three types: Malware, Personal Spyware, and Grayware. They evaluated the malicious mobile applications of different platforms, such as iOS, Symbian, and Android, and classified their malicious behavior as follows: Novelty and Amusement, Selling User Information, Stealing User Credentials, Premium-Rate Calls and SMS, SMS Spam, Search Engine Optimization and Ransom.

Enck, Octeau, McDaniel, and Chaudhuri [35] classified malicious behavior of mobile application into two categories as follows: information misuse and phone misuse , where information misuse means that privacy sensitive information on the devices (e.g. geographic location, IMEI, IMSI, ICC-ID, and etc.) has been being leaked outwards, and phone misuse means that the interface of the smart device has been manipulated in a wrong way, including telephony services, background recording of audio and video, socket API, and accessing the list of installed applications.

2.4.2 Analyzing Behaviors of Mobile Applications

There are two main approaches to analyze the behaviors of applications: the dynamic approach and the static approach, where the dynamic approach is used to perform the analysis through executing the application and the static approach is used to analyze the source or binary code of the applications without executions.

Enck, Gilbert, Han, Tendulkar, Chun, Cox, Jung, McDaniel, and Sheth [36] proposed a dynamic approach, TaintDroid for tracking and analyzing application behaviors in Android applications. TaintDroid automatically labels the privacy and sensitive data and tracks labels along with the propagation of the data through program variables, files, and interprocess messages. When the labeled data are transmitted via the Internet, or left to the system, TaintDroid would log the data's labels, the responsible application, and the destination where the data ate transmitted. It also produces real-time feedback to user so that user could know the security status on time. TaintDroid is limited to track data flows but not control flows.

Mann and Starostin [37] used static analysis to detect leakage of privacy data in Android applications. They sorted out private information into five categories: "location data", "unique identifiers", "call state", "authentication data", and "contact and calendar data". Their framework put signatures on the methods and parameters of the method which could be used to extract and transmit users private information private information. The framework was also restricted to track explicit information flow merely. The leakage of user's privacy data via implicit methods still could not be tracked.

Egele, Kruegel, Kirda, and Vigna [38] presented PiOS, the first static binary analysis that could analyze iOS applications, and automatically determine if these applications would leak user's private data. They decrypted binary of iOS applications, disassembled the binary to initialize the control flow graph of method calls of the binary, detected data flow analysis and found out potential privacy leaks. They evaluated the approach against more than 1,400 iPhone applications. This work briefly showed the feasibility of binary analysis for iOS applications.

Werthmann, Hund, Davi, Sadeghi, and Holz [39] proposed PSiOS, a framework for privacy data security. They constructed a protection layer between iOS runtime environment and the applications with MoCFI [40], which is a tool for iOS control flow integrity on the iOS platform.

Deng, Saltaformaggio, Zhang, and Xu [15] presented a hybrid approach that combines static and dynamic analyses. Their tool iRiS is capable of checking the abuses in private API in iOS applications. They used IDA Pro disassembler to generate control flow graph (CFG) on the target application, and then performed further analysis on the CFG. Once the static approach could not resolve call targets, they went on to use the dynamic approach to analyze the targets. This is done by extending the Valgrind [41].

We adopted the static approach presented in [42] to analyze the behaviors of iOS mobile apps.

The proposed risk evaluation techniques are orthogonal to behavior analysis techniques and can be integrated with other analysis tools as well for systematic recommendation.

3 Methodology

3.1 System Architecture

The architecture of AppReco is shown in Fig. 7. Before rating and recommending applications to the users, we could divide architecture into three parts: The first part, behavior analysis is from collecting application to checking application performing a certain behavior. It contains decrypting applications, disassembling applications, sensitive behavior lists, private API list, checking applications perform behaviors; The second part, application clustering is from gathering description to clustering applications with GHSOM clustering algorithm. It contains description preprocess, topic analysis with Latent Dirichlet Allocation, GHSOM clustering. The last part, rating sums the results before and gets recommendation apps. It contains our rating formula and computation processing. We would elaborate the three parts mentioned above in the following sections.

3.2 Same-Interest Application Discovery

Apple has divided applications into genres and sub-genres to help users to find out applications they interested in. Although genres could help users find out similar applications, some problems have arisen. For instance, some of the applications may share similarity with other applications but they are in different genres. In addition, if a genre is too general to effectively classify dimensional applications. There is a possibility for different types of applications labelled with same genres. Thus, we use machine learning clustering techniques to cluster the applications based on their descriptions. First, we use Latent Dirichlet Allocation algorithm to fetch the topics among applications and get the topic

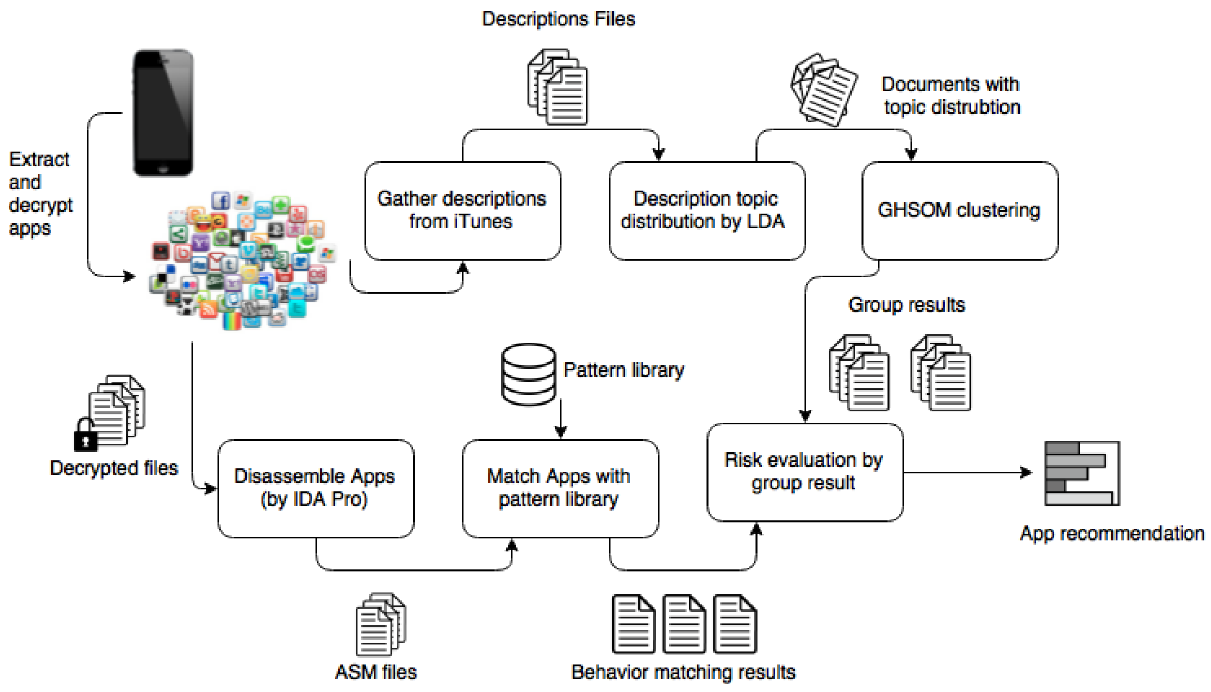

Descriptions Files
Documents with topic distrubtion
Extract and decrypt apps
Gather descriptions from iTunes
Description topic distribution by LDA
GHSOM clustering
Group results
Pattern library
Decrypted files
Disassemble Apps (by IDA Pro)
Match Apps with pattern library
Risk evaluation by group result
App recommendation
ASM files
Behavior matching results

Figure 7: Architecture of AppReco.

distribution about each application. By using the topic distribution as the input to GHSOM clustering algorithm, we can cluster the applications with similar topics. Finally, we get the clusters of applications sharing with the similar descriptions. We will introduce the clustering process in detail.

3.2.1 Using Descriptions to Discover App Interests

To discover app interests, we download the official description of the target app. This can be systematically crawled from Apple App Store. To extract the interests of apps, we perform the following preprocess operations. First, the descriptions of applications in App Store are often displayed in various languages, so we translate all the descriptions to English. We then are able to have the same base for word tokenizing, stemming and topic analysis afterwards. Word tokenizing removes punctuation marks and breaking sentences into words and elements. Word stemming, on the other hand, then removes morphological affixes from words so that only the word stems are left. For example, fight, fighting, fought, and fights would be reduced to the same stem, "fight". For each token in the list, we use NLTK package in python to do word stemming. After the stemming process, for each token, we further filter and delete *stop words*: words that are commonly appearing in documents without meanings, such as the, is, at, which,"who","I." The following is an example part from a weather application, Weather Motion after preprocessing description:

> weather motion take mobile weather apps whole new level stun high definition hd graphic sound effect see hear weather indoors head outdoors appadvice com feature macworld appguide re need soothe alarm clock way listen watch weather without go outside weather motion another way experience weather padvance com weather information style stylishipadapps tumblr comweather motion beautiful way check weather information iphone ipod touch application show local condition report top soothe video representation current weather condition description see current forecast weather gorgeous hd video high quality video footage different weather condition completely optimize iphone amaze retina display real time weather information base current location gps wifi wherever world local condition report temperature high low wind speed direction humidity level day local weather forecast every city slide finger screen see forecast temperature

Finally, we could generate a corpus from the list of processed descriptions. Then the corpus would be used as an LDA input to analyze the topics of whole descriptions in the following section.

3.2.2 Synthesizing Topic with LDA

To analyze sets of topics for the applications, we apply Latent Dirichlet Allocation (LDA) to discover topics. LDA is based on the hypothesis that each document could be viewed as a mixture of various topics. For example, the sentence,"learning English through sports communication", might be divided into language topics and sport topics in LDA model. Then LDA relies on statistical models to discover the elicit latent topics that occurs in a collection of unlabeled texts. A "topic" in LDA consists of a set of words that occurs together frequently. For instance, if we analyze lots of applications with sport genre with

LDA, the result would group words like "baseball", "fastball", "pitcher", and "catcher" into one cluster, and "basketball", "NBA", "dunk", and "FIBA" into another cluster. Some applications such as "MLB", "Fantasy Baseball" would be associated with the former topic, others like "NBA Game Time", "Pro Basketball Pocket Reference" would be assigned to the latter topic, and still others like "ESPN", "Yahoo Sports" would be connected to both topics. There would be another application not belonged to the two topics.

There are some different algorithms to implement LDA, such as EM algorithms and Gibbs Sampling. We choose the open-source LDA implementation - JGibbLDA as our tool [43]. LDA takes a corpus file and three input parameters:

(1) α, the number of latent topics to represent a document

(2) β, the number of terms to represent a latent topic

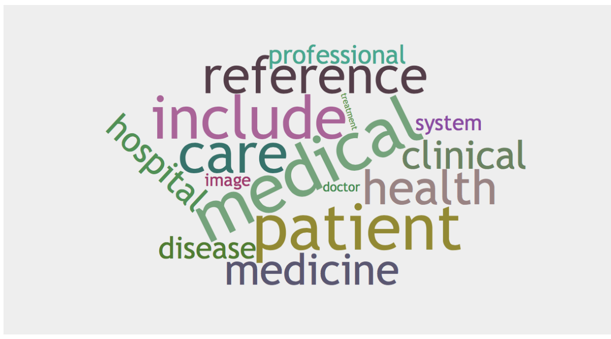
(3) The number of topics, the overall number of latent topics to be identified in the given corpus.

We have set the first two parameters by Griffiths and Steyvers' research [44] which discussed about choosing appropriate values for alpha and beta in LDA. The value of α is 50/the number of topics, and the value of β is 0.1. After some experiments with α and β, we fix the number of topics at a value of 25. With the value of parameters and the corpus which we have produced from the previous step(processed descriptions), we could discover the topics of apps and the topic distributions in these applications with LDA.

After LDA processing, we could collect the word distribution for each topic and the topic distribution for each document. The following Fig. 8 and Fig. 9 show two topics and their top terms as word clouds.

While the word distribution in topics show up, we could assign a name to each topic. For instance, we could assign "medical" topic to Fig. 8, and assign "calendar" topic to Fig. 9. Table 1 shows a set of topics assigned name and words of topics of one of the experiments.

At same time, we also retrieve topic distribution in each application. For instance,



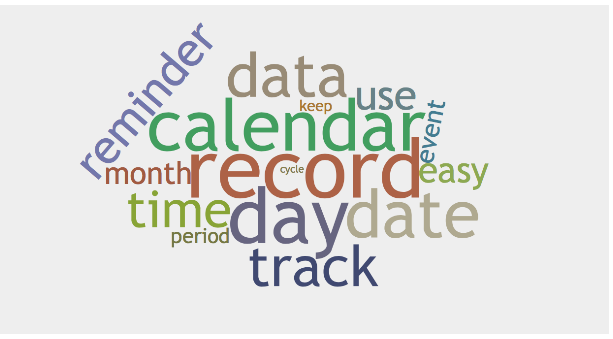


Figure 8: Sample topic #1 Figure 9: Sample topic #2

Table 1: Topics of one of the experiments

ID	Assigned Topic Name	Most Representative Words
1	iOS Device	iphone, ipad, support, touch, device
2	Video	video, watch, tv, news, live
3	Photo	photo, share, facebook, picture, friend
4	Design	color, design, create, app, tool
5	Language	word, english, learn, language, chinese
6	Fitness	time, weight, body, day, workout
7	Contact	service, card, phone, call, information
8	Free Application	app, free, com, get, apps
9	Weather	weather, time, day, location
10	Live Sports	game, player, play, live, score
11	Music	music, sound, song, play, sleep
12	Child	child, learn, fun, baby, kid
13	Food	food, recipe, store, eat
14	File Commander	support, file, download, view, share
15	Medical	medical, health, include, care, app
16	Love Story	life, name, good, love, people
17	Shopping	shop, buy, find, new, good
18	Newsstand	subscription, issue, purchase, period
19	News	read, news, new, magazine, content
20	Screen	version, screen, full, set, tap
21	Application	application, please, app, update
22	Taiwan Information	provide, book, taiwan, information
23	Calendar	contact, data, use, message, calendar
24	Finance	information, stock, service, market
25	Travel	map, travel, search, car, information

we could find a popular application named ”Google Translate.”. Its topic distribution is extremely high in topic #5, the language topic(62%) while no more than 5% in other topics. In contrast, some of applications cross two topics with both more than 10%. For example, there is an application named ”Fruits and vegetables flashcards quiz and matching game for toddlers and kids in English” assigned to the following two topics.

1. Topic 12 (Child) with a probability of 35.1%
2. Topic 5 (Language) with a probability of 12.1%,

By viewing its official description, we aware that the application could help child to learn the names of various fruits and vegetables, related to two different topics. The following is its preprocessed description:

> help child learn name various fruit vegetable encounter daily life use interactive picture book match game quiz game english language remember parent child first teacher app make child help discover learn name picture use interactive flash card different way feature beautiful eye catch picture professional pronunciation many language simple intuitive navigation apps flashcard match game quiza perfect kid book pronunciation voice early learn phone tablet app specifically design toddler baby mind simple intuitive navigation different picture feature fruit banana orange melon lime mango pineapple strawberry app real picture much easier baby relate compare draw animate image non native english speak app use teach child name common fruit vegetable thereby get good start learn english second language esl may also benefical child autisme asperges continuously expand range theme learn apps game child want get

3.2.3 Clustering Same-Interest Applications with GHSOM

After topic modeling with LDA, we could retrieve each application with probability of topic distribution, then the application could be identified by a vector of probabilities for each feature(topic). Then, we could do a clustering process to group applications with similar vector fields. We cluster applications with GHSOM algorithm, famous clustering algorithm for the feature of the dynamic and hierarchical structure. In AppReco, we use GHSOM in Matlab with GHSOM toolbox. We use GHSOM clustering to applications so we could get the proper size of group and the result of GHSOM is shown as Fig 10 and introduce the processing in detail in following paragraph.

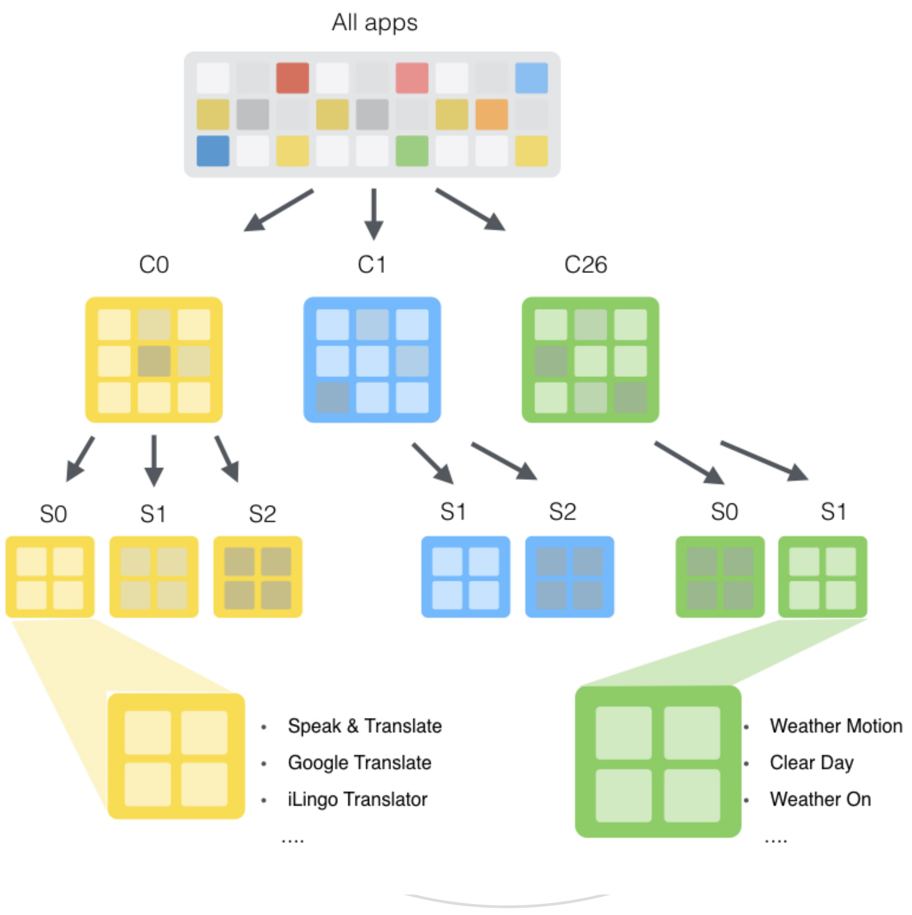


Figure 10: GHSOM tree structure for clustering applications.

1. We used τ_1 as 0.5 and τ_2 as 1.5 to GHSOM cluster algorithm.

2. Start at the layer 0, computing as the average of all input data as the weight vector and then calculate the mean quantization error of the map as mqe_0. The following formula is mqe_0 Eq.(4):

$$mqe_0 = \frac{1}{d}\|w_0 - x\| \tag{4}$$

In Eq.(4), d means the number of inputs data, m_0 represents the weight vector, and x is the value of the input data.

3. The train of the GHSOM starts with its first layer SOM, after computation of mqe_0. The first layer map would consist of a rather small number of units, such as a grid of 2×2 units.

4. Evaluate the result of SOM map, if each unit i of the map does not fulfill growing-stopping equation, it would continue growing to the next layer. The Eq.(5) is the stop growing formula.

$$mqe_i < \tau_2 \cdot mqe_0 \tag{5}$$

The Eq.(5), mqe_i means the value of mqe of the unit i. τ_2 represents the parameter to control the growth of GHSOM. If τ_2 is smaller, it is comparatively easy to subdivide to multiple layers.

5. When growing to next layer, GHSOM uses breadth parameter as τ_1 to control the growing map. GHOSM uses the Eq.(6) to calculate the mqe of the map, and then increases the nodes of the map until fulfilling the Eq.(7).

$$MQE_m = \frac{1}{\mu} \sum_i mqe_i \tag{6}$$

In Eq.(6), MQE_m represents the value of the map, while μ refers to the number of units i, contained in the map m.

$$MQE_m < \tau_1 \cdot mqe_0 \tag{7}$$

In Eq.(7), MQE_m means the value of mqe of the map, and τ_1 represents the parameter to control the growth of GHSOM. If τ_1 grows, MQE_m would increase positively. It would cause a relatively small number of the nodes of the map. In contrast, if τ_1 is reduced, MQE_m would decrease and then lead to a relatively large number of the nodes of the map.

6. Repeat step 4 to step 5 until all the units of map fulfill the Eq.(5).

The flexibility and hierarchical feature of GHSOM make it more capable of generating the hierarchical visualizing representation and tackling dynamic, growing and large data than SOM. With the result of the research, they proved GHSOM model able to uncover hierarchical structure. In recently years, GHSOM has been used in data mining, text mining and image recognition fields. Finally, we get lots of sub-clusters with fewer applications sharing more similar topics than the first clustering. We use this cluster result to make ratings and recommend applications to the user.

3.3 Behavior Analysis

AppReco uses static binary code analyzing check behaviors of applications. In order to analyze applications, we collect iOS applications from iTunes App Store, the official resource for iOS applications. However, downloading application is not enough. Apple has developed some security rules in the design of iOS so that users can not directly extract and analyze applications without root permission. Thus, we also decrypted applications to do binary code analyzing. We would discuss these parts in the following sections.

3.3.1 Identifying Sensitive Behaviors

We identify some behaviors in applications which are sensitive to users such as gathering calendars, reminders information and connecting the internet. We put these behaviors into the third categories: The first category is gathering and collecting privacy information of users through their devices (e.g. Photos, Reminder, Calendar, Location, Camera, Map, etc.); The second one is the usage of third party software development kit from famous social networking websites. It may gather user information from the websites, or post and share something on the websites. (e.g. Facebook, Google, etc.). The other one is promoting advertisement for users (e.g.Google AD, iAD, etc.). The detail of sensitive behavior lists in iOS 9 would be shown in Table 2.

We list some common framework and class to achieve these behaviors as patterns, and

Table 2: Sensitive behavior lists in iOS 9

Behavior	Framework List
User's Privacy Information	Bluetooth, Calendars and Reminders, GameKit, HealthKit, HomeKit, Location, Microphone, MapKit, Photos and Camera, StoreKit
Third Party SDK	Faceook, Google, Twitter
Advertisement	iAD, GoogleAD, AmazonAD

then get these framework from iOS developer library and other third party framework. There is a sample pattern to promote advertisement with iAD framework shown in Fig. 11.

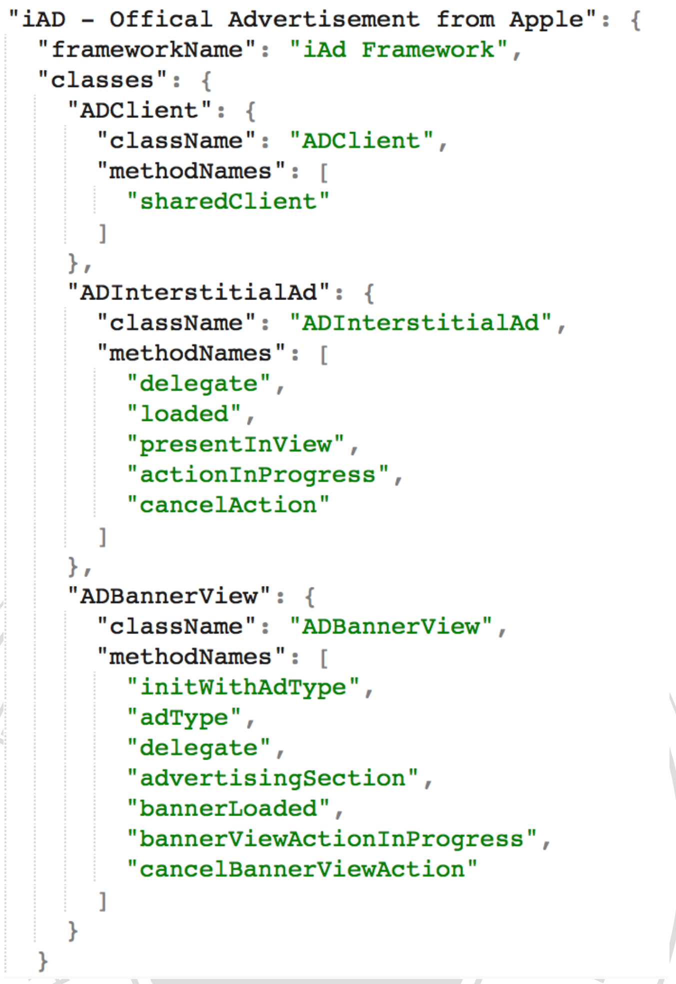
3.3.2 Collecting Private API

AppReco not only check the behaviors from public API and third party frameworks but also investigate applications for the usage of private API. Private API is only used for Apple's built-in applications and public frameworks because it could provide powerful functionality that could threaten the security of the system. Some method in private API such as [AADeviceInfo appleIDClientIdentifier] which could obtain the Apple ID of the device user, [AADeviceInfo serialNumber] which could obtain the serial number of the device, and so on. We extract private APIs from the dynamic shared libraries in the iOS software development kit. There are 372 private API frameworks in iOS 9.0 version and among the 21,159 different classes. There is a sample one of private API gathering detail information of devices in iOS 9 shown in Fig. 12.

If an application detected the usage of private API, it would not be not recommended to users during the later rating and ranking process.

3.3.3 Checking Behaviors

In this section, we scan the application ASM files with behavior patterns and private API lists. ASM file is written in assembly language, a low level programming language that can be converted to machine language. We could find out class, method names in the application. If the occurrences of a set of the certain function call, defined as the pattern, matches with the target application, it's possible that the target application may



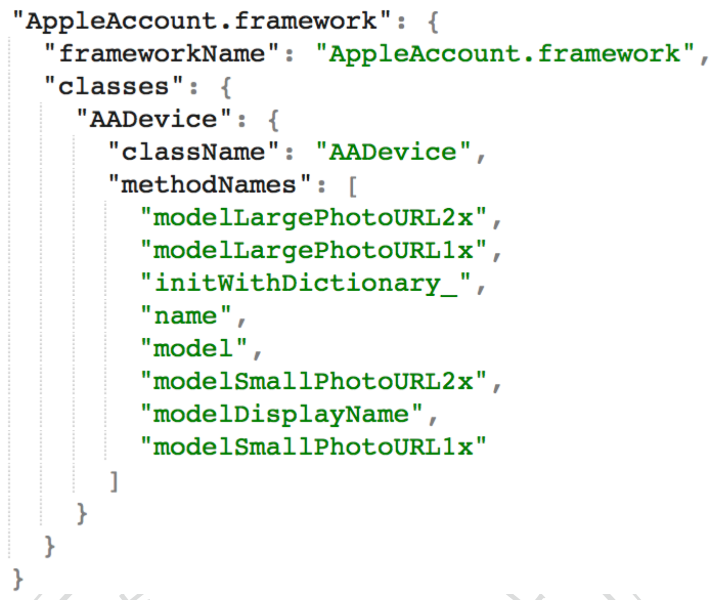
```
"iAD - Offical Advertisement from Apple": {
  "frameworkName": "iAd Framework",
  "classes": {
    "ADClient": {
      "className": "ADClient",
      "methodNames": [
        "sharedClient"
      ]
    },
    "ADInterstitialAd": {
      "className": "ADInterstitialAd",
      "methodNames": [
        "delegate",
        "loaded",
        "presentInView",
        "actionInProgress",
        "cancelAction"
      ]
    },
    "ADBannerView": {
      "className": "ADBannerView",
      "methodNames": [
        "initWithAdType",
        "adType",
        "delegate",
        "advertisingSection",
        "bannerLoaded",
        "bannerViewActionInProgress",
        "cancelBannerViewAction"
      ]
    }
  }
```

Figure 11: Sample patterns for advertisement implement on iOS 9

contain the certain pattern and have the possibility to perform a certain behavior. If an application has the possibility to practice a certain behavior, AppReco would give it 1 score on the behavior. In contrast, if an application doesn't have the possibility to practice the behavior, AppReco would give it 0 score on the behavior. These ratings would be weighted in the rating section.

3.4 Recommendation

To evaluate our behavior-aware method for application recommendation, we apply a ranking formula for application recommendation. With ranking equation, we could suggest user applications from behaviors of applications.



```
"AppleAccount.framework": {
  "frameworkName": "AppleAccount.framework",
  "classes": {
    "AADevice": {
      "className": "AADevice",
      "methodNames": [
        "modelLargePhotoURL2x",
        "modelLargePhotoURL1x",
        "initWithDictionary_",
        "name",
        "model",
        "modelSmallPhotoURL2x",
        "modelDisplayName",
        "modelSmallPhotoURL1x"
      ]
    }
  }
}
```

Figure 12: One of private API in iOS 9

3.4.1 Concept of AppReco Recommendation

AppReco would calculate application behavior score for users. AppReco collects sensitive behavior patterns such as using a calendar, location, Bluetooth, and etc., and then detects what kinds of behavior an application would perform. Then we would use weighting function to adjust the score. The weighting function not only represents the behavior of an application but also reflects differences within the applications in a same group and the applications in a dataset. AppReco uses the idea of term frequency and inverse document frequency statistic method for our risk evaluation.

The detail concept illustration is in Fig. 13. First, we find out that behavior 1 takes an important role in app 1. It means that behavior 1 is more important than other behaviors in app 1. We could use the term frequency in this part to emphasize which behavior is more important than other behaviors in an app. Second, we could discover that the usage of behavior 3 is rare in cluster 1. In contrast, the usage of behavior 4 is common in cluster 1. We could use the inverse document frequency in this part to reflect that app uses a rare or common behavior in this cluster. Finally, we could see that behavior 1 is widely used in many apps, it means that behavior 1 is a popular behavior. We could use the inverse document frequency in this part to reflect if an app uses a rare or common behavior in the dataset.

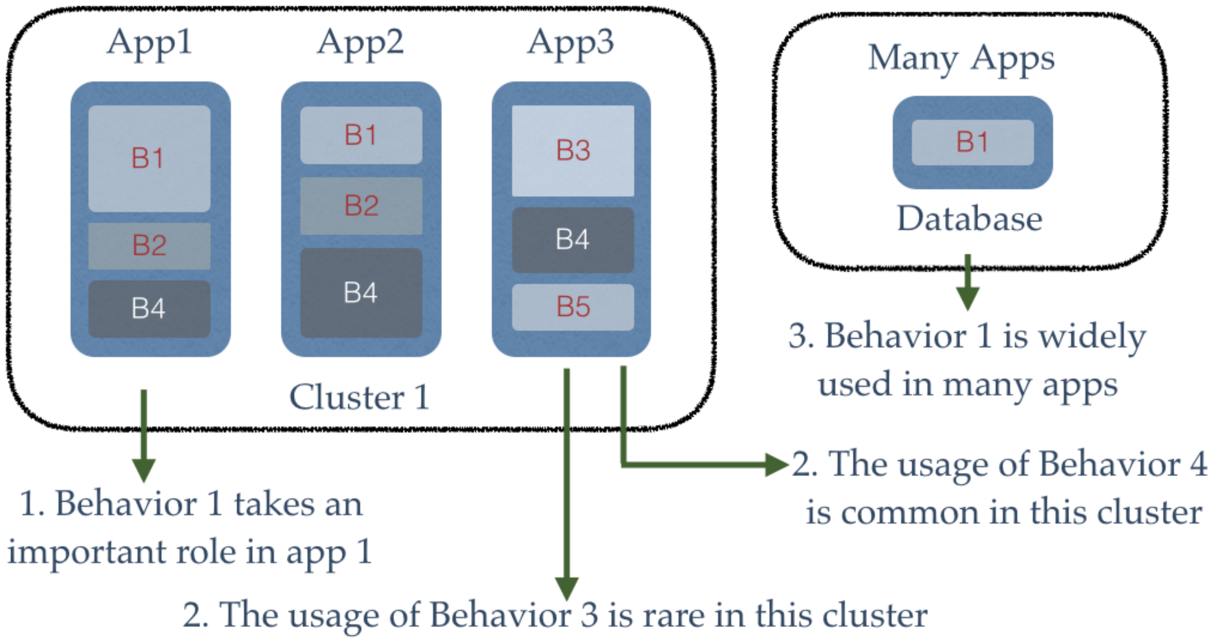

App1
App2
App3
B1
B2
B4
B1
B2
B4
B3
B4
B5
Cluster 1
Many Apps
B1
Database
3. Behavior 1 is widely used in many apps
2. The usage of Behavior 4 is common in this cluster
1. Behavior 1 takes an important role in app 1
2. The usage of Behavior 3 is rare in this cluster

Figure 13: The concept of recommendation of AppReco

TF-IDF approach can reflect how important a word is to a document in a collection or corpus. Therefore, TF-IDF approach is widely used in text mining and information retrieval field. There are several variants of TF and IDF weight schema, and different combination between TF and IDF can reach different results. Salton and Buckley [45] used several different data collections (ACM, TIMES, INSPEC, NPC, 12864 docs (largest at that time)). They proposed the best TF-IDF weighting combination as Eq.(8):

$$w_{i,d} = \left(0.5 + \frac{0.5tf_{i,d}}{\max\ tf_d}\right) \cdot \left(\log\left(\frac{N}{df_{i,D}} + 1\right)\right) \tag{8}$$

It could be separated into two parts by multiplication dot. The first part is TF weighting function. $tf_{i,d}$ means term frequency, the number of times that a term i appears in a document d. max tf_d is the most common term times of a document d. It uses 0.5 as normalization vector and raw frequency, divided by the maximum raw frequency to prevent a bias toward longer documents. The second part is IDF weighting function. $df_{i,D}$

means the number of documents which contain a specific term i and N is the total number of documents. Afterwards, we use logarithm to scale down the number to avoid a huge number. In the meantime, to avoid $df_{i,d}$ equals 0, which would lead to a division-by-zero, we adjust the denominator to add 1 to the equation.

3.4.2 Recommending Based on Ranking Formula

AppReco calculates the three types of risk by the corresponding categories of behaviors. Then, it would sum the score of three risks as the total risk. The higher the risk an application gets, there is more possibility of performing sensitive behavior in the application. AppReco also highlights applications which are detected the usage of private API. Once an applications private API calls are revealed, it would not recommend it to user. AppReco adjusts Eq.(8) to calculate behavior score. The following formula is our behavior score Eq.(9).

$$w_{b,a} = \left(0.5 + \frac{0.5bf_{b,a}}{\max\ bf_a}\right) \cdot \left(\log\left(\frac{N_C}{df_{b,C}} + 1\right)\right) \cdot \left(\log\left(\frac{N}{df_{b,A}} + 1\right)\right) \qquad (9)$$

First, AppReco reflects app behavior score on how many function calls under the behavior pattern. AppReco uses TF weighting function to distinguish behaviors which take an important role in application, and then it replaces $tf_{i,d}$ with $bf_{b,a}$. It means behavior frequency, the number of times that a behavior b appears in an application a. max bf_a is the most common behavior times of an application a.

Second, AppReco compares an application's behavior with other applications in the same cluster as a weighted basis. AppReco applies IDF weighting function to measure whether a behavior is rare or common in the cluster. N_C means the number of applications in this cluster, and $df_{b,C}$ means the number of applications, which behave in the cluster. We could get the number which presents the behavior risk in the cluster by dividing N_C by $df_{b,C}$. If there is a behavior which is seldom adopted in the cluster, it means that the behavior is not really required in the cluster. Thus, an application in the cluster, which executes the seldom-performed behavior, would receive greater number. In contrast, if there are lots of applications practicing the same behavior in the cluster, it means that this behavior is required in the cluster. An application in the cluster performing the common behavior would receive lower number.

Finally, AppReco performs IDF weighting function the second time to check whether the behavior is rare or not. N is the number of applications in AppReco, and $df_{b,A}$ is the number of applications performing a certain behavior in AppReco. If there is the majority of applications using the behavior in AppReco, it implies that the behavior is widely-used by developers, so the potential risk evaluation would be low. On the contrary, if there is the minority of applications using the behavior, it indicates that the behavior is used by a small number of developers, so the potential risk evaluation would be comparatively high.

4 Evaluation

We have implemented the tool AppReco that integrates our previous work AppBeach [42]. We report some analysis results against iOS 9 applications in this section. The complete analysis result can be found online [46].

4.1 Data Set

We have collected around 6840 iOS applications, covering over twenty genres such as Social Networking, Utilities, Reference, etc. from the official App Store. The complete data can be found in our website [46].

We download these applications from iTunes top free and top grossing charts, that categorize applications. In this way, it is easy to reach famous and popular applications. For example, "SoundCloud" is at the top of music genre charts. "Google Translate" is at the top of reference genre charts. "Pic Collage" is at the top of photo & videos genre chars. The extracted topics are listed in Table 3.

Table 3: Topic result in iOS 9

ID	Assigned Topic Name	Most Representative Words
1	Information	information, provide, service, user, market
2	Music	music, sound, play, song, feature
3	Design	color, use, create, design, screen
4	Sport	live, sport, event, app, team
5	Language	word, english, learn, language, chinese
6	Fashion	love, friend, fashion, face, beauty
7	Video	video, support, watch, tv, view
8	Children Book	book, story, child, first, read
9	Children Game	fun, game, child, learn, kid
10	Food	shop, food, recipe, find, store
11	Life	life, name, good, people, day
12	Exercise	body, weight, workout, exercise, time
13	Weather	weather, time, location, information, day
14	Recommend	app, get, make, best, like
15	Game	game, can, play, level, get
16	Subscription	subscription, issue, purchase, period, account
17	Apple	iphone, device, use, ipad, contact
18	Baby	application, baby, design, product, group
19	News	news, read, content, new, ipad
20	Travel	map, travel, car, offline, information
21	Health	medical, include, system, care, health
22	Time	record, use, time, data, list
23	Finance	service, card, provide, information, bank
24	Photo	photo, share, file, image, support
25	Free	app, com, free, please, http

Then, we apply GHSOM for a two-layer-clustering. We used τ_1 as 0.5 and τ_2 as 1 to GHSOM cluster algorithm. Each cluster can be indexed with C-S, where C indicates the cluster in the first layer and S indicates the sub-cluster of the C cluster. After GHSOM processing, we have generated 27 clusters, and 1100 sub-clusters. We apply the static behavior analysis on apps within the same cluster and also rank apps within the same cluster to recommend less risky apps to users.

The following is some clusters including famous applications, that can serve as the examples to demonstrate our analysis results.

4.2 The cluster of Clear Day

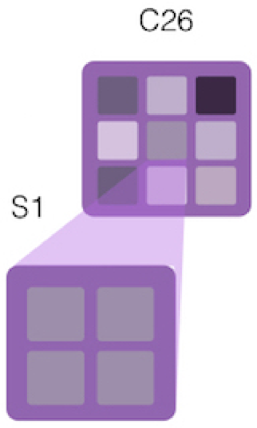


Figure 14: Cluster 26-1

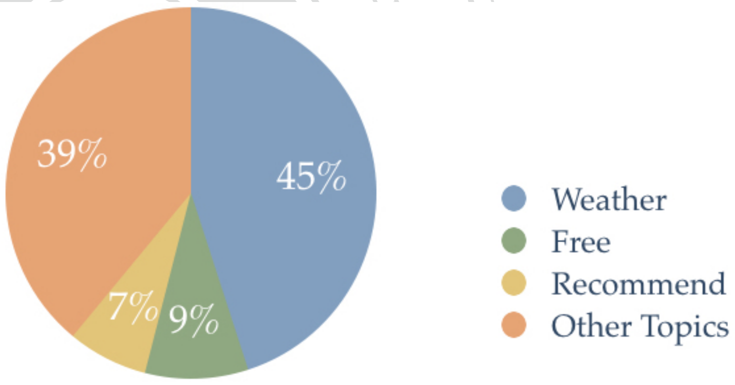


Figure 15: The topic distribution of cluster 26-1

This cluster is mainly composed of Weather topic (45.3%), "Free" topic (8.5%) and other topics are lower than 8%. The most famous application in this cluster is "Clear Day - Weather HD - Live Weather Forecast with NOAA Radar Free", which provides users with weather information through real weather condition videos. Other applications in Table 4 are also weather or weather-related applications.

After detecting the behaviors of these applications, we find out that "Clear Day" uses more functions than the others. For example, it uses the third party package, "Facebook" function calls.

Among thus cluster, we recommend the weather applications, "wetter.com Weather HD Lite" and "Weather On - Push Notification" to users because they use less function calls and have similar functionality.

Table 4: The cluster of Clear Day

App ID	App Name	Genre	Privacy	3rd Party	Ad.	Total	pAPI
412489722	Clear Day - Weather HD - Live Weather Forecast ...	Weather	2.0	2.92	0.67	5.59	-
554438153	Weather On - Push Notification	Weather	0.8	0.0	1.11	1.91	-
711410889	Weather Mate - Live Current Conditions ...	Weather	2.28	1.61	0.66	4.55	-
379403071	wetter.com Weather HD Lite	Weather	2.62	0.0	0.0	2.62	-
476662435	Weather Motion Free	Weather	4.25	0.0	1.11	5.36	-

"App ID" indicates an application ID that uniquely identifies an iOS application.

"App Name" indicates an application name.

"Genre" indicates an application category that gives by developers.

"Privacy" indicates the total point gained from the privacy behavior category.

"3rd Party" indicates the total point gained from the third party SDK behavior category.

"Ad" indicates the total point gained from the advertisement behavior category.

"Total" indicates the total point summed of "Privacy", "3rd Party" and "Ad" points.

"pAPI" indicates the usage of "Private API".

4.3 The cluster of Google Translate

"Language" topic (60%) is the dominant topic in this cluster while other topics take the proportions less than 5%. The most popular application is "Google Translate", which is an application for translating within languages. There are also other translator applications, for example, "PONS Online Dictionary - Your free online translator" is an application for language translator, and "iLingo Translator" is a well-known application

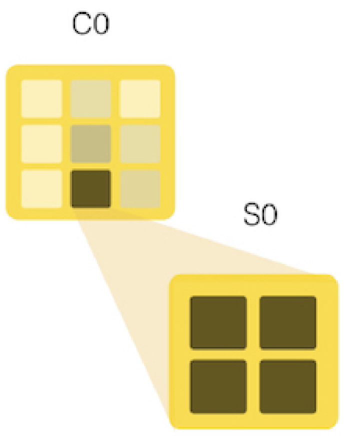


Figure 16: Cluster 0-0

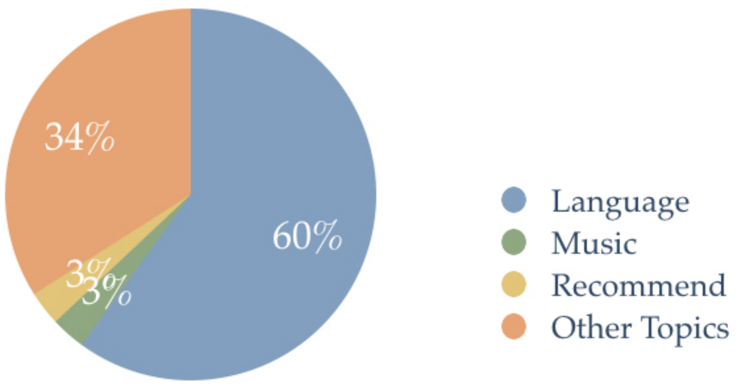


Figure 17: The topic distribution of cluster 0-0

for text translator with speech. Other applications in Table 5 are dictionary or translator applications. Interestingly, in this cluster, some applications actually have similar functionality, but they are labelled with different genres in Apple App Store.

In this cluster, on the one hand, the applications with higher risk are "iLingo Translator" and "PONS Online Dictionary". The former is risky for its privacy information and advertisement, when it uses location data and microphone for translator with speech and prompt advertisement to users. The latter one is risky for its usage of the third party SDK, through which it practices a sharing function, allowing users to share the application information with others through Twitter, Facebook, messages, and etc..

On the other hand, we recommend "Google Translate" and "Speak & Translate - Free Live Voice and Text Translator with Speech" to users because they use less function calls and have similar functionality in this cluster.

Table 5: The cluster of Google Translate

App ID	App Name	Genre	Privacy	3rd Party	Ad.	Total	pAPI
804641004	Speak & Translate - Free Live Voice and Text ...	Productivity	0.0	0.0	0.0	0.0	-
338687553	iPronunciation free - 60+ languages ...	Education	0.11	1.83	0.3	2.24	-
577741918	PONS Online Dictionary - Your free online ...	Reference	1.29	2.12	0.18	3.59	-
414706506	Google Translate	Reference	0.22	0.0	0.0	0.22	-
683978652	Crazy English Listening & Speaking Tutorial ...	Sports	0.26	0.0	0.3	0.56	-
407952605	CamDictionary Free	Reference	1.0	0.0	0.0	1.0	-
492870498	LearnChinese Free - Find Chinese ...	Business	0.26	0.0	0.18	0.44	-
624487979	Talk!Talk! Korean Word Book-Basic	Education	1.79	1.13	0.18	3.1	-
646665738	iLingo Translator - Translate Text to Speech for ...	Reference	3.15	0.0	2.01	5.16	-

4.4 The cluster of SoundCloud

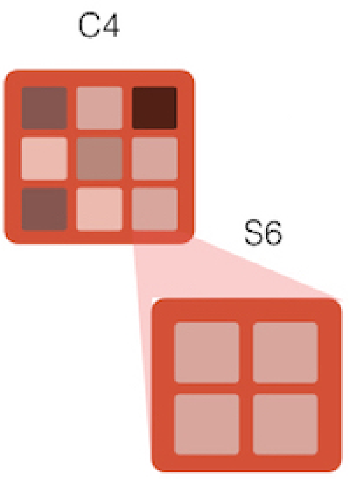


Figure 18: Cluster 4-6

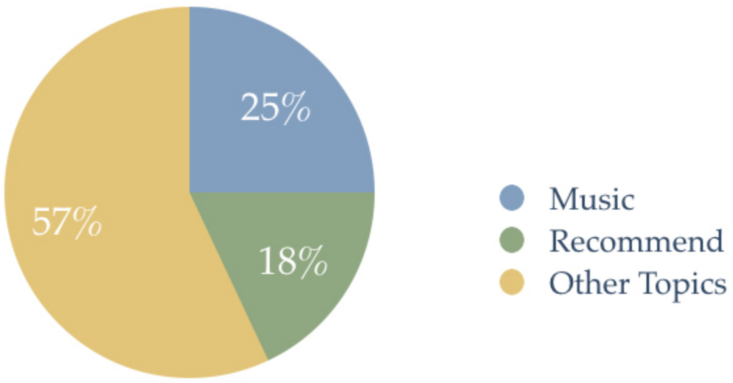


Figure 19: The topic distribution of cluster 4-6

The topic distribution of cluster has a large proportion in "Music" topic (24.8%) and "Recommend" topic (18%). We infer that the applications in this cluster are highly music-related. Among this cluster, the most popular application is "SoundCloud - Music & Audio", which is for listening to music and discovering new audios. Other music-related applications are also found among the cluster, for example, "Sound Sleeper" is an application to offer relaxing sound and white noise to users. "Crossfader" is an application for playing music like DJ. Other applications in Table 6 are also music player or relaxing music applications.

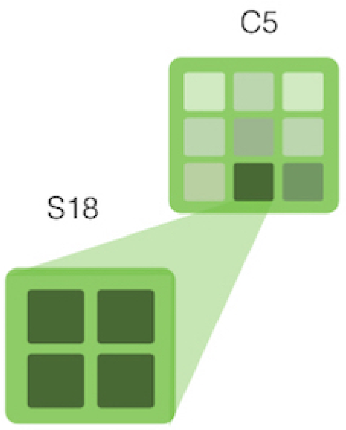
Among the cluster, the riskiest application is "Music top 100's hits". It is the only application, that uses "Calendars and Reminders", the official built-in framework in Apple, which allows an application to grant access to user's calendar.

In contrast, we recommend "SoundCloud" and "Naturespace - Relax Meditate Focus Sleep and Rest with 3D Sounds, sonic therapy for anxiety and stress relief" to users because they use less function calls and have similar functionality.

Table 6: The cluster of SoundCloud

App ID	App Name	Genre	Privacy	3rd Party	Ad.	Total	pAPI
312618509	Naturespace - Relax Meditate Focus Sleep ...	Health & Fitness	1.31	0.0	0.0	1.31	-
555564825	Sleep Time: Sleep Cycle Smart Alarm ...	Health & Fitness	0.9	1.13	0.0	2.03	-
391732937	Music top 100's hits	Music	4.2	2.13	1.31	7.64	-
628517073	Crossfader	Music	1.37	0.0	0.0	1.37	-
540360389	nana - sing, record and share your music!	Music	2.17	0.0	0.0	2.17	-
336353151	SoundCloud - Music & Audio	Music	0.0	0.0	0.0	0.0	-
778773354	Sound Sleeper - calming, soothing sounds of ...	Health & Fitness	1.39	1.13	0.0	2.52	-

4.5 The cluster of PicCollage



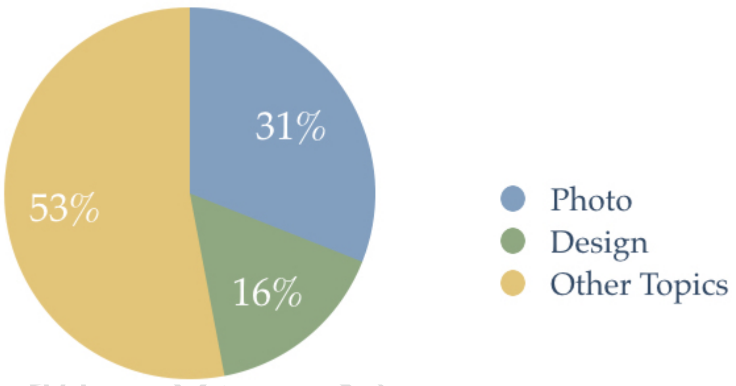


Figure 20: Cluster 5-18

Figure 21: The topic distribution of cluster 5-18

The topic distribution in the cluster are rather high in ”Photo” topic (31.3%) and ”Design” topic (16%). Thus, we infer that the applications in this cluster are mainly about editing photos. The most famous application is ”Pic Collage”, which is for photo editor, that imports pictures from users’ photo library, Instagram, Facebook and then shares revised works to Instagram, Facebook, Twitter. Other applications in Table 7 are picture or sticker editor applications. They have a wide usage of behavior, photo and camera, because most of them need to import pictures from the device storage. Some of them practice the third party SDK, so they can use sharing functions to share photos through social networks.

Among the cluster, the riskiest application is ”Video Collage - Free vid frame creator for instagram, and youtube.” It is the only application that uses the behavior,”Calendars and Reminders”, the official built-in framework in Apple, which allows an application to grant access to user’s calendar.

We recommend photo editor applications, ”Pic Stitch - #1 Photo and Video Collage Maker”, ”CollageIt Free - An Automatic Photo Collage Maker” to users because they use less function calls and have similar functionality.

Table 7: The cluster of PicCollage

App ID	App Name	Genre	Privacy	3rd Party	Ad.	Total	pAPI
509987785	Pic Jointer Picture Collage, Camera ...	Photo & Video	0.75	0.71	0.33	1.79	-
575442281	CollageIt Free - An Automatic Photo ...	Photo & Video	0.64	0.72	0.0	1.36	-
454768104	Pic Stitch - #1 Photo Collage	Photo & Video	0.26	0.7	0.0	0.96	-
523481129	Whitagram	Photo & Video	0.85	0.0	0.0	0.85	-
712175767	Video Collage - Free vid frame creator for ...	Photo & Video	1.99	0.7	0.33	3.02	-
448639966	Pic Collage - Photo collage editor with ...	Photo & Video	0.75	0.7	0.0	1.45	-
665832780	Nice Day Stamp by PhotoUp - Cute and ...	Photo & Video	0.8	0.61	0.55	1.96	-
546616462	WhatsGoLa Free	Photo & Video	1.68	0.71	0.33	2.72	-

5 Conclusion

5.1 Conclusion

We propose behavior-aware recommendation for iOS mobile applications, integrating static binary analysis for behavior analysis with topic and app clustering for finding similar interest apps. We rank apps according to the analysis of their embedded behaviors in executable, and hence are able to discover the similar-interest app that has less dislike behaviors to users. We implement the tool AppReco and report our preliminary results against thousands of online iOS applications. We show that without expertise and user experiences, we are able to provide users mobile applications that match their interests with less undesired behaviors.

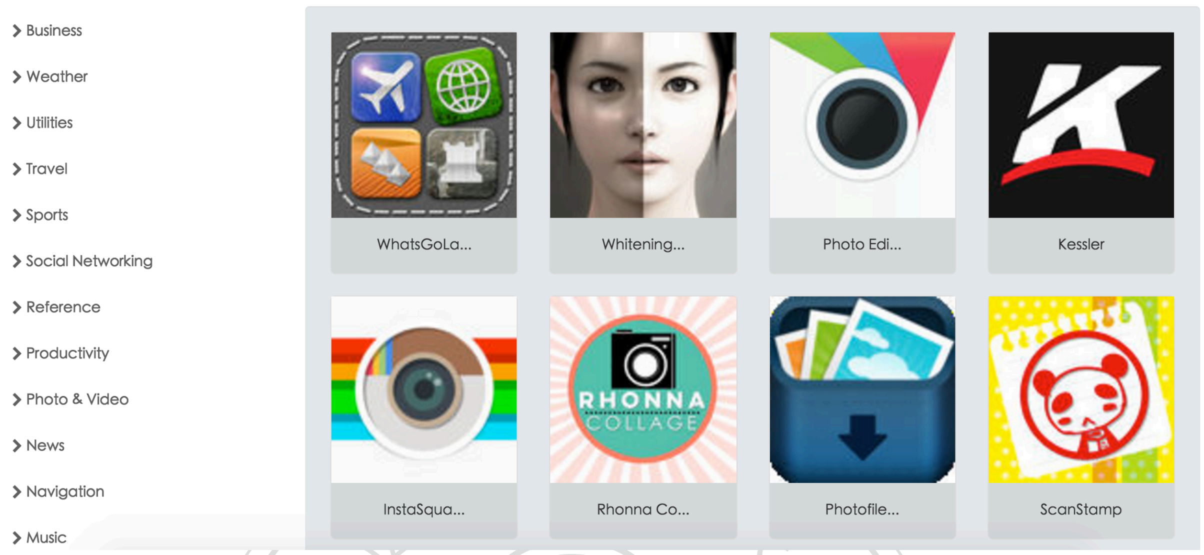
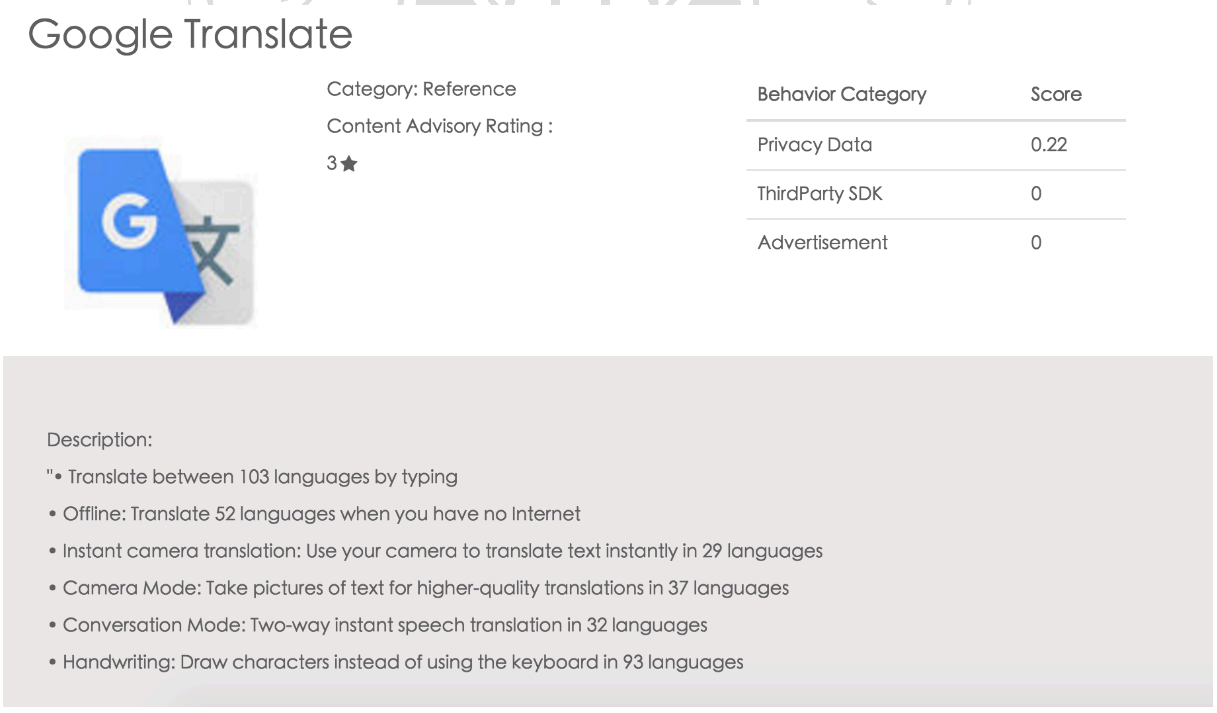


Figure 22: AppReco Website Index Page



Google Translate

Category: Reference

Content Advisory Rating :

3★

Behavior Category	Score
Privacy Data	0.22
ThirdParty SDK	0
Advertisement	0

Description:

"• Translate between 103 languages by typing

• Offline: Translate 52 languages when you have no Internet

• Instant camera translation: Use your camera to translate text instantly in 29 languages

• Camera Mode: Take pictures of text for higher-quality translations in 37 languages

• Conversation Mode: Two-way instant speech translation in 32 languages

• Handwriting: Draw characters instead of using the keyboard in 93 languages

Figure 23: AppReco Website App Page

5.2 Limitation

There are three limitations of the presented approach. First, we have to get an iOS application executable, that is available for the analysis. It is necessary to jailbreak the device in order to retrieve app executables; hence the latest apps that are not runnable on jailbroken devices may not be analyzed with the presented approach. For this reason, the subject of the reported experiments is highly limited to iOS version applications. Second, we can only conduct the behavior, that is included in the framework. If the developer performs a similar behavior which uses other frameworks or third party packages, we would not be able to check it. However, if the frameworks are included among our pattern, it could be detected. Third, we cluster these applications according to their descriptions. As a result, if the description of an application is too unambiguous to correctly represent its features, it is possible that it would be categorized into some actually dissimilar applications.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[2] "Number of available applications in the google play store from december 2009 to november 2015." `http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/`. (Visited on 02/22/2016).

[3] "Number of available apps in the apple app store from july 2008 to june 2015." `http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/`. (Visited on 02/22/2016).

[4] "Apps4review.com." `http://apps4review.com`. (Visited on 01/04/2016).

[5] B. Yan and G. Chen, "Appjoy: personalized mobile application discovery," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp. 113–126, ACM, 2011.

[6] "The sweet setup." `http://thesweetsetup.com`. (Visited on 01/04/2016).

[7] "Path app under fire for unauthorized address book upload." `http://appleinsider.com/articles/12/02/07/path_app_under_fire_for_unauthorized_address_book_upload.html`. (Visited on 01/04/2016).

[8] "G data mobile malware report threat report: Q3/2015." `https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/G_DATA_MobileMWR_Q3_2015_EN.pdf`. (Visited on 01/04/2016).

[9] "Mcafee labs threats report november 2015." `http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-nov-2015.pdf`. (Visited on 01/04/2016).

[10] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pp. 620–629, IEEE, 2005.

[11] A. Beach, M. Gartrell, and R. Han, "Solutions to security and privacy issues in mobile social networking," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 4, pp. 1036–1042, IEEE, 2009.

[12] "Mobilead2013." `http://www.emarketer.com/Article/Driven-by-Facebook-Google-Mobile-Ad-Market-Soars-10537-2013/1010690`. (Visited on 01/04/2016).

[13] "Gartner says mobile advertising spending will reach $18 billion in 2014." `http://www.gartner.com/newsroom/id/2653121`. (Visited on 01/04/2016).

[14] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond, "Truth in advertising: The hidden cost of mobile ads for software developers," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp. 100–110, 2015.

[15] Z. Deng, B. Saltaformaggio, X. Zhang, and D. Xu, "iris: Vetting private api abuse in ios applications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 44–56, ACM, 2015.

[16] "ios developer library." `https://developer.apple.com/library/ios/navigation`. (Visited on 01/04/2016).

[17] "nst/ios-runtime-headers." `https://github.com/nst/iOS-Runtime-Headers`. (Visited on 01/04/2016).

[18] "Appbrain." `http://www.appbrain.com`. (Visited on 01/04/2016).

[19] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pp. 175–186, ACM, 1994.

[20] M. J. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & webert: Identifying interesting web sites," in *AAAI/IAAI, Vol. 1*, pp. 54–61, 1996.

[21] R. Van Meteren and M. Van Someren, "Using content-based filtering for recommendation," in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, pp. 47–56, 2000.

[22] M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[23] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

[24] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52, Morgan Kaufmann Publishers Inc., 1998.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, ACM, 2001.

[26] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[27] T. Hofmann, “Probabilistic latent semantic analysis,” in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 289–296, Morgan Kaufmann Publishers Inc., 1999.

[28] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent dirichlet allocation for tag recommendation,” in *Proceedings of the third ACM conference on Recommender systems*, pp. 61–68, ACM, 2009.

[29] T. Hofmann, “Collaborative filtering via gaussian probabilistic latent semantic analysis,” in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR ’03, (New York, NY, USA), ACM, 2003.

[30] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, “Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences.,” in *ISMIR*, vol. 6, p. 7th, 2006.

[31] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks," *Int. J. Approx. Reasoning*, vol. 51, no. 7, pp. 785–799, 2010.

[32] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, and R. Van de Walle, "Using topic models for twitter hashtag recommendation," in *Proceedings of the 22nd international conference on World Wide Web companion*, pp. 593–596, International World Wide Web Conferences Steering Committee, 2013.

[33] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychological review*, vol. 104, no. 2, p. 211, 1997.

[34] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pp. 3–14, 2011.

[35] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security.," in *USENIX security symposium*, vol. 2, p. 2, 2011.

[36] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, p. 5, 2014.

[37] C. Mann and A. Starostin, "A framework for static detection of privacy leaks in android applications," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 1457–1462, ACM, 2012.

[38] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications.," in *NDSS*, 2011.

[39] T. Werthmann, R. Hund, L. Davi, A.-R. Sadeghi, and T. Holz, "Psios: bring your own privacy & security to ios devices," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 13–24, ACM, 2013.

[40] L. Davi, A. Dmitrienko, M. Egele, T. Fischer, T. Holz, R. Hund, S. Nürnberger, and A.-R. Sadeghi, "Mocfi: A framework to mitigate control-flow attacks on smartphones.," in *NDSS*, 2012.

[41] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," in *ACM Sigplan notices*, vol. 42, pp. 89–100, ACM, 2007.

[42] F. Yu, Y.-C. Lee, S. Tai, and W.-S. Tang, "Appbeach: Characterizing app behaviors via static binary analysis," in *Proceedings of the 2013 IEEE Second International Conference on Mobile Services*, p. 86, IEEE Computer Society, 2013.

[43] "Jgibblda:a java implementation of latent dirichlet allocation (lda) using gibbs sampling for parameter estimation and inference." `http://jgibblda.sourceforge.net`. (Visited on 01/04/2016).

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

[45] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[46] "Appbeach." `http://soslab.nccu.edu.tw/appbeach`, 2014. (Visited on 01/04/2016).