ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Classifying the Mobile Apps with Risk Score by using the Enriched Information of App Context

Lokhande Prajakta Padmakar, Shivaji R. Lahane

PG Student, Department of Computer Engineering, Affiliated to Savitribai Phule Pune University, GES's R. H. Sapat College of Engineering, Management Studies & Research, Nashik, India

Assistant Professor, Department of Computer Engineering, Affiliated to Savitribai Phule Pune University, GES's R. H. Sapat College of Engineering, Management Studies & Research, Nashik, India

ABSTRACT: Increased craze of smartphones has opened a huge market for android applications. As a result large numbers of mobile apps are coming into the market by multiple vendors, out of which most of the apps are designed to perform same functionality. This has raised two problems: first the applications present in the market are not being properly categorized. Classifying theses apps properly can be useful for various factors like making it time efficient and easy for the user while selecting the required app, to understand the user preferences that can motivate the intelligent personalized services. But this requires a detailed information about the app, which is a nontrivial task as limited information about the apps is directly available. Second it is found that the permissions the app request at the time of installation can be used to expose users sensitive and private information to the outside world without his knowledge, if the app is having some malicious intention. This is possible as most of these apps come from an unknown vendor and so there is higher possibility of them being malicious. To solve these two problems, we have developed a system in which we have considered different sources for collecting the information about the app like information from the labels (app name), from search engine, contextual usage history of the app collected from the users usage record and the permissions of the app, which they request at the time of installation giving us a secure and effective classification of the apps. We have compared our results with the exiting categories of the apps given on app store, it provides appropriate results with our defined categories. Our system also provides a facility to recommend the apps to the user based on the user preferences.

KEYWORDS: Mobile apps, Classification, Security, Risk score, Enriched information

I. INTRODUCTION

Today, with the evolution in the mobile technology the mobile devices have turned into a ubiquitous device not limited only to calling and texting but provide different kind of facilities like camera, email facility, access to social network, video calling etc. This has given a rise to a new market, providing huge amount of mobile apps to provide different services. Also the easy to understand, tools present today have made it possible for anyone with little knowledge to design the mobile apps. It is found that many of these apps have same functionality. As a result having a classification of these apps will play an important role not only to the user in order to search the required app easily but also for analysing the user preferences which can help the intellectual services like app recommendation, target advertising, user segmentation etc [1].

Effective classification plays a very important role for having a proper mobile app usage analysis. Classification of these mobile apps is considered as a quite difficult task, because for having a proper or effective classification we need to have detailed information about the app. This is challenging task as the contextual information available about the app is very limited this is because the words used for app name are very short and sparse. The objective of this paper is to design a system that will provide an effective classification of the mobile apps by using the enriched information about the apps.

To achieve this goal, we not only consider the name of the app but also the web based information and the real world contextual features about the app. This will improve the contextual information of the apps, resulting into an improved performance of the classification. Here the web based information is extracted from the general search engine like google or from the app store and the real world features will be extracted from the mobile usage record of the user.

Along with this information, the list permissions requested by the app at the time of installation are also extracted. This is because it is found that malicious attack is possible by the apps using these permissions i.e. through these permissions, apps can get access to user's private and sensitive information. The current security mechanism provided by android cannot detect such malicious apps, as the security mechanism provided by android is in standalone fashion [2] i.e. the user is supposed to make the decision about accepting app, by allowing the app to access the resources requested after reading the permissions requested by that app. This is easily ignored by the users, as they mainly focus on the reviews and ratings, at the time of selecting the app. So by exploiting these permissions and calculating the risk score based on them, provide us a more accurate and secure classification of the mobile apps

II. RELATED WORK

Our problem to automatically classify the mobile apps can also be considered as a problem to classify the short and sparse text. Here we have considered the existing work done first with regards to the classification of the short and sparse texts and then the earlier work done for improving the security of the android system.

A. *Classification of short and sparse text:*

X. H. Phan et al [3] presented a general framework to process the short and sparse text documents on the web. They have focused mainly on the data sparseness and synonyms/hyponyms by exploiting the hidden topics discovered from large scale external document collection i.e. using the universal data set. Exploiting the hidden topics has helped to improve the representation of the short and sparse text for the classification. The semantic topics are the additional textual features they have integrated with the words to improve the classification. Sahami and T.D. Heilman [4], presented a similarity kernel function based approach to find the similarity between the short text. They found that the traditional cosine similarity measure produces inadequate results. They proved that there approach can effectively measure the similarity between short text snippets which by exploiting the web search engine provide greater context for the short texts. A.Z. Broder et al [5] proposed a methodology to classify these short queries using blind feedback technique i.e. after a query is given its topics are determined from the web searched results that are returned for the query. Authors proved that the methodology yields higher classification for the queries with help of the empirical evaluation performed. H. Ma, et al [6], proposed an approach which leverages search snippets to build vector space for both app usage and categories and classify the app usage records using the cosine space distance. H. Zhu, H et al [7] proposed an approach to classify the mobile app using the enriched information for that purpose the authors have first exploited the web based features of the app from the web search engine, but as the information obtained is limited and then from the observation that different types of mobile apps can be relevant to different real-world contexts, they have extracted some contextual features for the mobile apps from the mobile users context-rich device logs, but in this only the explicit contextual features of the mobile apps are taken into consideration.

B. *Android Security:*

Applications before installation ask for the permissions to access some or the other kind of information from user's device. Unknown to the user these apps may get access to some sensitive information present on the users device, which might be harmful to the user in case the malicious apps uses this information for their beneficial purpose. In order to make the user aware about what kind of data is being accessed by the apps they have installed, W.Enack et al [7], proposed one tracking system for having a real time privacy monitoring on the smart phones. Which will inform the user when the application is trying to send sensitive data from the phone. But it lacks at the time of defending against the security and monetary focused malware which send out spam or create premium SMS messages without accessing private information. Applications when downloaded from the smart phones requests for permission which the user need to accept before they are being downloaded. A.P. Felt et al [9] studied these permissions and came to conclusion that most the apps asked for large number of permissions, which they don't even require for their processing. This makes the apps more threatening as the attacker through these apps may try to get access to the sensitive information of the user. Authors here have used the static analysis to determine the over privileged apps, on a

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

set of 940 applications and find that about one-third are over privileged. E. Chin et al [10] conducted one user study to understand user's perceptions of smart phone security and installation habits. Where they found that the users don't focus on the permissions at the time of app browsing and installation, they mostly relay on the user rating and reviews while selecting the app. B.P. Sarma et al, [11] investigated the feasibility of apps permissions with respect to their category without considering the usefulness of the app.

III. PRE-REQUISITES

Here some prerequisites are being defined which we use in to design the system.
Search snippets: this is the information obtained from the web search engine after the name of the app is submitted to the search engine. Search snippets provide us with more relevant words with respect to the app name we have given. We use this information where we extract the web based features for classifying the app. We use the google search engine to obtain the snippets for the app and consider only the top results obtained. Here top sixteen results out of the obtained results are being considered.

Context usage records: these are the records obtained from different people, which gives the information about their app usages like which apps are being mostly used by the people and for what amount of time. This data is obtained from the app which tracks the user's usage record and is being used in the phase where we extract the contextual features of the app. We also maintain personalized

Permission records: these records are obtained by extracting the manifest file of the mobile apps, as it is not possible to obtain information about the permissions the app requests at the time of installation directly. We use these records to calculate the risk score of the apps.

Latent Dichcrit Allocation(LDA): For learning the latent semantic topic, LDA model is being used. It is basically a generative model used for collections of grouped discrete data. Where each group is considered as a random mixture over a set of latent topics and where each topic is discrete distribution over the collections vocabulary. To train the LDA model, Gibbs sampling is being used[12].

Maximum entropy model: It is a machine learning model used for probabilistic classification belonging to the class of exponential models. It is based on the principal of maximum entropy; we use this classifier in the training phase to combine the different features extracted for the app. It can be explained in general using three steps as[14] :

- For every word *w* and class *c* $\in C$ *where* C is the predefined dataset, define a joint feature *f*(*w*, *c*) = *N* where *N* gives the number of times that *w* has occurred in a document for class *c*.
- Through iterative optimization, a weight is assigned to each joint feature so as to maximize the log-likelihood of the training data.
- The probability of class *c* given a document *d* is given as:

$$P\,(c|\,d) = \frac{1}{Z(d)}\ \exp\left(\textstyle\sum_i \lambda i f i(d,c)\right) \qquad (1)$$

Naïve bayes model:
It is also a machine learning model similar to that of MaxEntropy model. But in this it assumes that the features are conditionally independent of each other. It is based on the bayes theorem, and is being used in the final phase while calculating the risk score of the app. It is explained in generalized form as[13]:

$$P\,(c|\,x) = \frac{P(c|x)P(c)}{P(x)} \qquad (2)$$

Where ,
P(c|x) is the posterior probability of class (target) given predictor (attribute).
P(c) is the prior probability of class.
P(x|c) is the likelihood which is the probability of predictor given class.
P(x) is the prior probability of predictor.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

IV. Proposed System

We have developed a system using web services and mobile app to provide recommendation to the user based on his preferences and get the detailed statistical analysis of the requested app. Architecture of the system is shown in Fig 1, according to which our system consists of a mobile application for user that provides user interface to the user and the whole app classification process is carried out at the server end. Server side consists of three web services namely web based information, context based information and respond to user request. Using the first two web services information for classifying the app i.e wed based and context based is obtained respectively. The third web service is used to communicate with the mobile app, through which the server responds to the user requests. Database is present in which the information about the app details is being stored and administrator is present to manage the system. The working of the system is divided in different modules as shown in Fig 2.

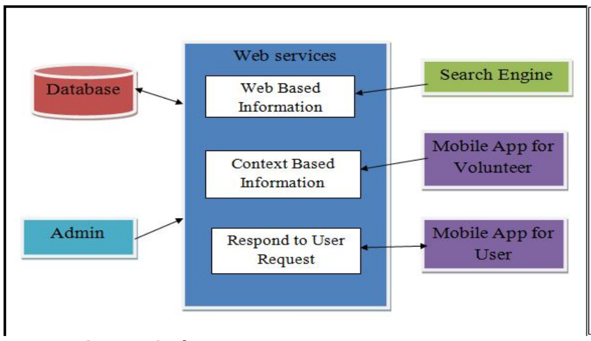


Fig 1. System Architecture

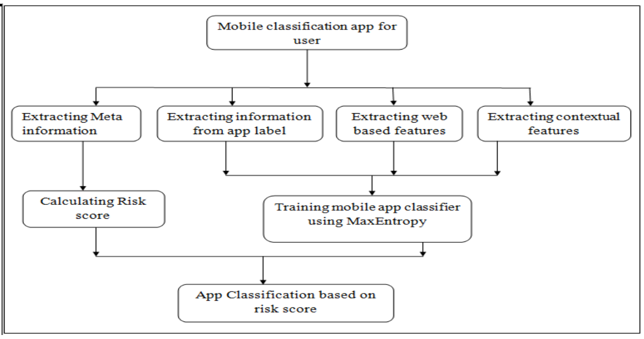

Mobile classification app for user
Extracting Meta information
Extracting information from app label
Extracting web based features
Extracting contextual features
Calculating Risk score
Training mobile app classifier using MaxEntropy
App Classification based on risk score

Fig 2. Different modules of the system

A. *Module 1: training the app classifier using the using MaxEntropy classifier:*

For training the MaxEntropy classifier we are giving the features collected from different sources as an input. The classifier will be trained using the algorithms present to train the machine learning model. So that when the app will be given to the classifier it will effectively classify it into the category the app will belong. The features are obtained include both explicit and implicit features of the app.

1) Web based feature extraction:

Here the features extraction takes place with the information obtained from the web search engine. Both the explicit and implicit features are obtained as explained below.

a) Explicit web based feature extraction

Here the explicit features of the app will be extracted from web search engine. In simple words after giving the name of the app to the system, it will extract the top results from the search engine to place the app in the appropriate category. To achieve this goal vector space model will be used. Which consist of the three steps, according to which first, one category profile d_c will be build by integrating all the M snippets retrieved for some app. Here the stop words will be removed and the verbs and adjectives will be normalized Then in the second step normalized word vector will be build for each app category $\overrightarrow{\boldsymbol{w_c}}$ = dim [n] using the following formula (3):

$$\dim[i] = \frac{freq_{i,c}}{\sum_i freq_{i,c}} \quad (1 \le i \le n) \qquad (3)$$

Where, $freq_{i,\,c}$ is the frequency of ith word in the category profile. Similarly, in the last step for each snippet s retrieved for the app a we will build word vector $\overrightarrow{w_{a,s}}$ and calculate the cosine distance between $\overrightarrow{w_c}$ and $\overrightarrow{w_{a,s}}$. From the result obtained, we will consider the max similarity result and assume that category to be the appropriate category for app a.

$$c^{*} = \arg\max \text{Similarity}\,(\overrightarrow{w_{a,s}}, \overrightarrow{w_c}) \qquad (4)$$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Then to confirm our result we calculate the general label confidence score by (5):

$$GConf(c,a) = \frac{M_{c,a}}{M} \qquad (5)$$

Where, $M_{c,a}$ is the number of returned related search snippets of app a, whose category labels are c after mapping.

b) Implicit web based features extraction

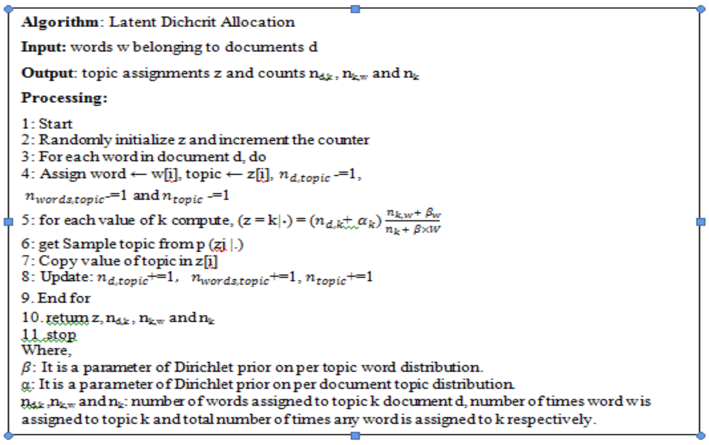
Here the latent semantic meaning of the snippets retrieved will be considered; as a result this will give us a more refined result. This is because in the explicit feature selection the latent semantic meaning of the words is not considered i.e. for example the words like ``play", ``game" and ``funny" are considered as totally different while calculating the distance between the word vectors. But after considering their latent semantic they can be placed in the same semantic topic of ``entertainment". To understand the latent semantic meaning we will use the latent dirichlet allocation (LDA) model. Basic algorithm for the LDA is as shown in Fig 3. Using this model we will build a category profile d_c. Then for given app a top M result will be retrieved and the words which do not match with category pro.file will be removed. Then KL divergent will be used to map each snippet with the category. The formula for KL-Divergent is given by (6):

$$D_{KL}(P(z|s)||P(z|c)) = \sum_k P(z_k|s) ln \frac{P(z_k|s)}{P(z_k|c)} \qquad (6)$$

Where, z is the latent topic of the category. The category with the smallest KL-divergence is selected. To confirm our result we then calculate the topic confidence score for the given category as follows (7):

$$TConf(a,c) = \frac{T_{a,c}}{M} \qquad (7)$$

Where, $T_{a,\,c}$ is the number of returned snippets of app a with respect to category label c. The topic confidence score gives the confidence that app a is labelled as category c with respect to the latent semantic topic.



Algorithm: Latent Dichcrit Allocation

Input: words w belonging to documents d

Output: topic assignments z and counts $n_{d,k}$, $n_{k,w}$ and n_k

Processing:

```
1: Start
2: Randomly initialize z and increment the counter
3: For each word in document d, do
4: Assign word ← w[i], topic ← z[i], $n_{d,topic}$ -=1,
 $n_{words,topic}$-=1 and $n_{topic}$ -=1
5: for each value of k compute, (z = k|·) = ($n_{d,k}$+ $\alpha_k$) $\frac{n_{k,w}+\beta_w}{n_k+\beta\times W}$
6: get Sample topic from p (zi |.)
7: Copy value of topic in z[i]
8: Update: $n_{d,topic}$+=1,  $n_{words,topic}$+=1, $n_{topic}$+=1
9. End for
10. return z, $n_{d,k}$, $n_{k,w}$ and $n_k$
11 stop
```

Where,
β: It is a parameter of Dirichlet prior on per topic word distribution.
α: It is a parameter of Dirichlet prior on per document topic distribution.
$n_{d,k}$, $n_{k,w}$ and n_k: number of words assigned to topic k document d, number of times word w is assigned to topic k and total number of times any word is assigned to k respectively.

Fig3. Algorithm for LDA

2) Context based feature extraction:

In this we consider the context feature of the app from real world context log. Here also we consider the explicit and implicit features.

a) Explicit contextual feature:

In this the real world context information is collected from different user's logs in terms of feature-value pair. In simple words for an app belonging to some category we consider when the app is being used and how many time it is being used. A context profile R_a is build by using these feature-value pair, for each app. Similarly we also build a context profile for each category R_c, by combining the context profiles of the pre-selected apps labelled with c. After that a vector is build for the app and category and we calculate the cosine distance between the context vector of the app and context vector of all the categories. At last the cosine distance calculated is arranged in descending order and the one with max similarity is assumed as the category. The result is then confirmed by calculating category rank distance given by (8):

$$CRDistance\,(a,c) \;=\; Rk(c) - 1 \qquad (8)$$

Where, Rk(c) is the rank of the category c, which is obtained by comparing the vector distance to app a. Here smaller the distance more accurate is our selected category.

b) Implicit contextual feature:

In implicit feedback the semantic meaning of the contextual feature-value pairs will be obtained from the explicit contextual features available. Like for example feature-value pairs like "Time-span: 2 hours", "age: 15-20" can be grouped together under the topic "trendy app". To achieve this goal here we category profile R_c is build using Latent Dirichlet Allocation on context model. Accordingly, for a given app a, category profile Ra will be build using the historic logs database. Then KL-Divergence will be calculated between each category and the app using the given formula (9)

$$D_{KL}(P\,(z|\,Ra)\;||\;P\,(z\,|\,Rc)) = \sum_k P\,(z_k\,|Ra)\,\ln\frac{P(z_k|Ra)}{P(Z_k|Rc} \qquad (9)$$

From this the smallest KL-Distance is considered as the category for the app and finally, topical Rank distance will be calculated to confirm the result. The topical rank distance is calculated by:

$$TRDistance\,(a,c) \;=\; Rk(c) - 1\,1 \qquad (10)$$

B. *Module 2:*

In this module we calculate the risk score of the apps. Here the risk score is being calculated on the basis of the permissions the app requests at the time of installation. We consider the permissions to be app category specific. Once the permission information is obtained then using the rarity of the critical permission we have calculated the risk score of the app using the naïve bayes classifier, steps of which are shown in Fig 5. Permissions are extracted from the manifest file of the app using the steps shown in Fig 4.

C. *Module 3:*

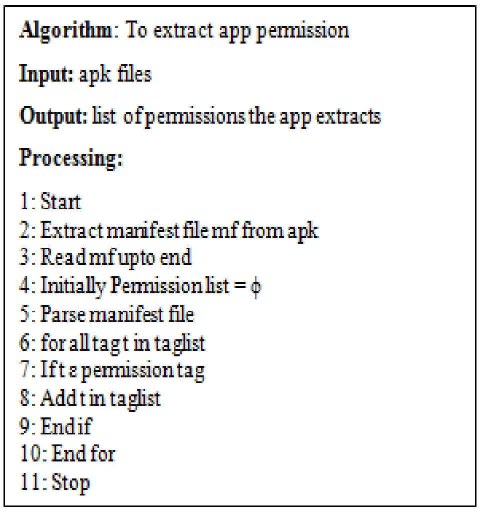
This is the final stage in which we combine the results obtained from the first two modules. So after the app will be classified into its specific category in module 1 and its risk score is being calculated in module 2 we combine these two results to give us the proper information about the app. We have also extracted other details for the app from the play store like its current version, total number of downloads, rating etc. By combining this information with our obtained category and risk score gives us detailed statistical information of a particular app. Then using the contextual information obtained from the user, app is being recomendended to him by comparing his information with the information that is present in the database. The recommendations here are generated using the MaxEntropy classifier.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

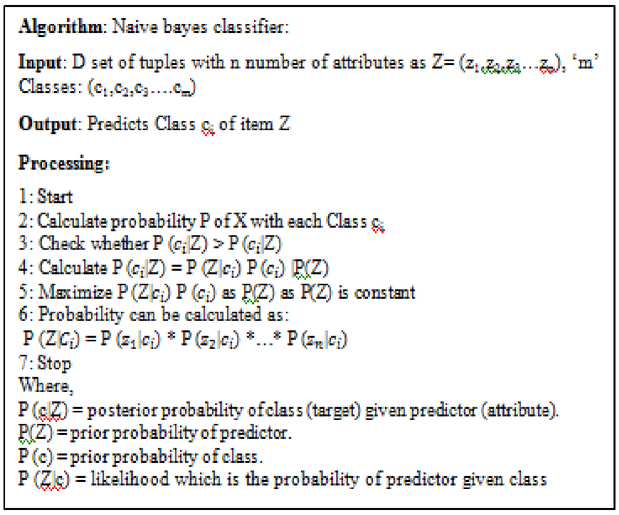
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015



```
Algorithm: To extract app permission
Input: apk files
Output: list of permissions the app extracts
Processing:
1: Start
2: Extract manifest file mf from apk
3: Read mf upto end
4: Initially Permission list = ϕ
5: Parse manifest file
6: for all tag t in taglist
7: If t ε permission tag
8: Add t in taglist
9: End if
10: End for
11: Stop
```

Fig 4.Steps to extract permissions from apk file



```
Algorithm: Naive bayes classifier:
Input: D set of tuples with n number of attributes as Z= (z1,z2,z3…zn), 'm' Classes: (c1,c2,c3….cm)
Output: Predicts Class ci of item Z
Processing:
1: Start
2: Calculate probability P of X with each Class ci
3: Check whether P (ci|Z) > P (ci|Z)
4: Calculate P (ci|Z) = P (Z|ci) P (ci) |P(Z)
5: Maximize P (Z|ci) P (ci) as P(Z) as P(Z) is constant
6: Probability can be calculated as:
 P (Z|Ci) = P (z1|ci) * P (z2|ci) *…* P (zn|ci)
7: Stop
Where,
P (c|Z) = posterior probability of class (target) given predictor (attribute).
P(Z) = prior probability of predictor.
P (c) = prior probability of class.
P (Z|c) = likelihood which is the probability of predictor given class
```

Fig 5. Steps for naive byes classifier

V. EXPERIMENTAL SETUP

For performing the experiment as mentioned above we have developed web services and mobile application. For creating the web services we have used java and apache server, mobile app is created using android sdk in eclipse. This apk communicate with the web service using http protocol. We have used a mobile app called app usage tracker to obtain the context information from different people, by installing this app on their device. This information is exported in xls file. This file and user personal information is uploaded as contextual information of user. The format of xls file is, this xls is pre-processed and statistical information is which includes the name of the app, time duration for which the app is being used like morning, afternoon, evening. The user app and the user response web service communicate with each other, in which the user will get the analytical result of the app requested by the user. Administrator is present to maintain the server system. All the results obtained are stored in the database, which is created using the wamp server.

Dataset:

Multiple datasets are being used to perform the experiment as explained below:

- **App Category**: We have created 2 level category set. The level-1 contains 11 category lists. Each category is the subdivided in to level-2 categories. We have crated 44 such categories sample of which is shown in Table 1.
-

Level-1 Categories	Level-2 Categories
Internet	Web browser , etc
Business	Office Tools , Security , etc
Communication	Call , Mail and SMS , etc
Game	Action , adventure , etc
Multimedia	Audio , Video , etc
Navigation	City Guides , Maps , etc
SNS	Facebook,Tweeter , etc
System	Management , Performance , etc
Reference	News , Utility , Reading , etc

Table 1. App category

- **List of Category keyword:** this list contains category level-2 specific keyword list <catid, keyword list ,wt>. Sample of the list of category keywords is as shown in Table 2.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

category	keyword list
action games	running, jumping, fighting, heroes, etc
board games	Board, players, dice, chances, etc
Business - office tools	manage, file, word, excel, pdf, etc
business-job apps	jobs ,search, company, etc
Reference-Dictionary	thesaurus, antonyms, synonyms, etc

Table 2. Keywords with respect to category

- **Permission Dataset**: This data set contains level-2 category wise permission details. We create this dataset programmatically. This can be recorded as: <category, permission name, class, risk score>
- **Context log**: This dataset contains user specific app access information in the form <uid, app name, access details> . Then the dataset of the users in the format <name, age, occupation, gender>

VI. RESULTS

In this section we present the results obtained for various test cases. We are testing our system on windows 7 os for the server side and the app developed on the Sony xperia phone using different mobile apps present on the play store. The results obtained for the tested apps for different test cases are as follows

1. Total number of words present in the app name

Here we from the tested mobile apps we find the total number of words present in the app name. in the result it is found that most of the apps are having only two words. Which proves that it is difficult to obtain more information from the app name to categorize the app. The result is shown in Fig 6. Where x-axis is the number of words and y axis is the number of apps

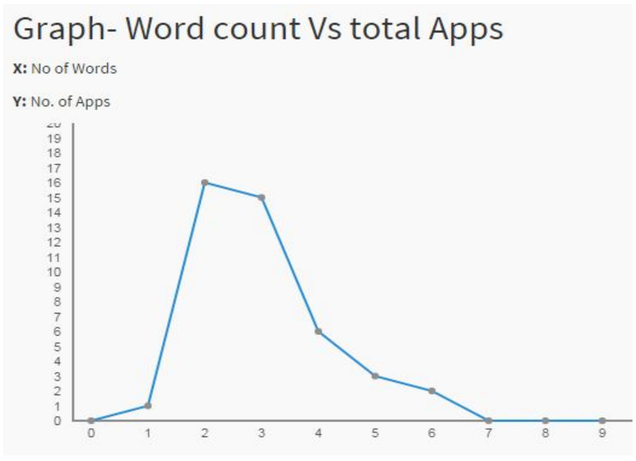


Fig.6 word count vs total app

2. Number of apps matching with level 2 category:

Here we have tested the number of apps matching to our defined level2 category, i.e. the name of the app directly matching with the name of the category. The result is shown in Fig 7, where y axis is the number of apps tested and x-axis is the total number of category.

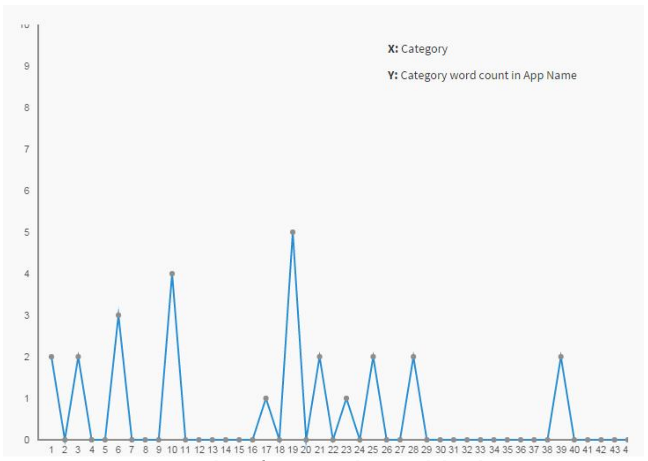
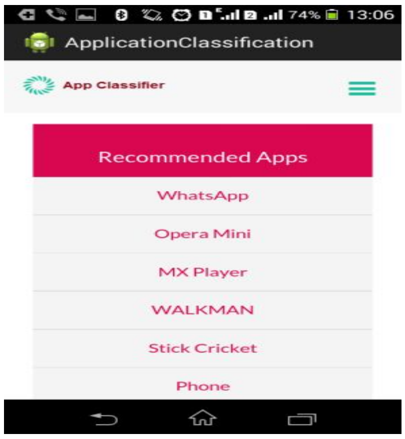


Fig.7 Number of apps matching with level 2 category

3. Recommendation generated for the user based on his profile created.

Here we test the apps recommended to the user based on his profile being created. We have shown result for the user who has entered the profile information as age between 19-30, occupation as student, gender as female. Based on this information the apps recommended through the app are shown in Fig.8.


ApplicationClassification

App Classifier

Recommended Apps

WhatsApp

Opera Mini

MX Player

WALKMAN

Stick Cricket

Phone

Fig. 12 Recommended apps

4. Categorization of the app into particular category:

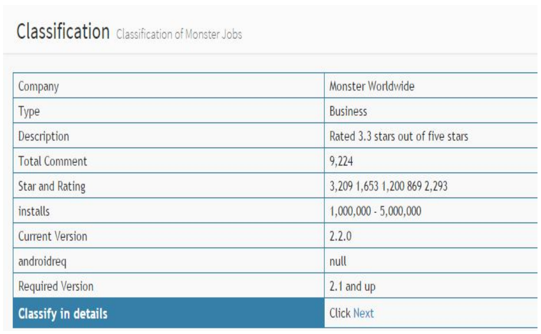
In this we show how a particular app is being categorized by our system. After the name of the app is given based on the information obtained the app is being categorized. We have shown the result by taking example of one job app called “monster”. Fig 9 shows the information about the same app shown on the play store; here we have extracted the information of the app present on the play store. Giving us the detail about the category in which the “monster” app is being placed, i.e. “business” category.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

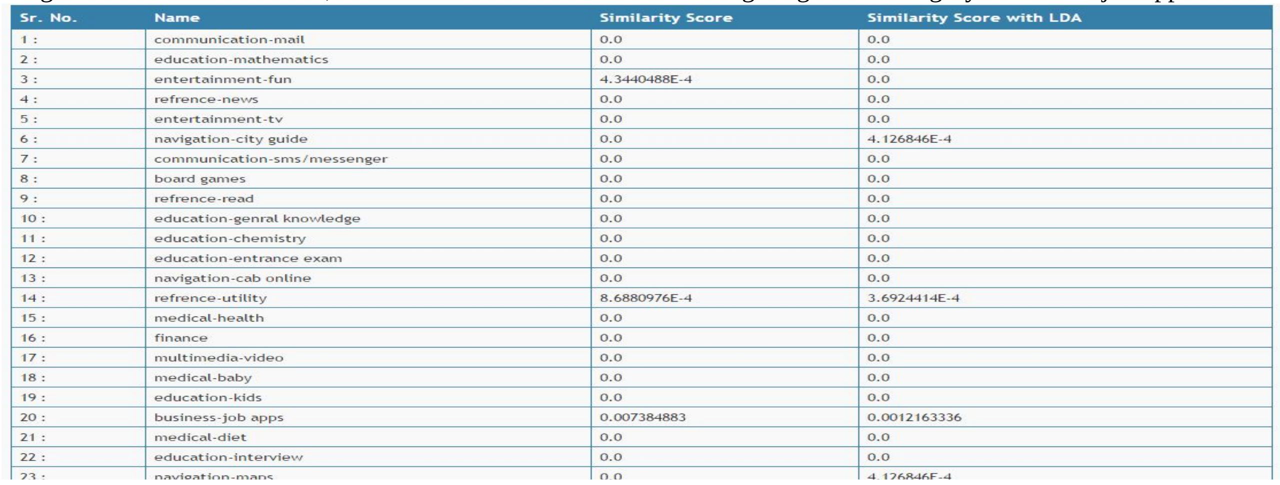


Classification Classification of Monster Jobs

Company	Monster Worldwide
Type	Business
Description	Rated 3.3 stars out of five stars
Total Comment	9,224
Star and Rating	3,209 1,653 1,200 869 2,293
installs	1,000,000 - 5,000,000
Current Version	2.2.0
androidreq	null
Required Version	2.1 and up
Classify in details	Click Next

Fig 9. Result obtained from play store

Our system then calculates the confidence score for both explicit and implicit method i.e. without and with LDA respectively, without LDA the keywords are being directly matched with the information. With LDA matching is done based on the weights given to the keyword. Fig 10 shows the confidence scores obtain for the app for different categories. Based on this score, the maximum score is considered i.e. giving us the category "business- job apps".



Sr. No.	Name	Similarity Score	Similarity Score with LDA
1 :	communication-mail	0.0	0.0
2 :	education-mathematics	0.0	0.0
3 :	entertainment-fun	4.3440488E-4	0.0
4 :	refrence-news	0.0	0.0
5 :	entertainment-tv	0.0	0.0
6 :	navigation-city guide	0.0	4.126846E-4
7 :	communication-sms/messenger	0.0	0.0
8 :	board games	0.0	0.0
9 :	refrence-read	0.0	0.0
10 :	education-genral knowledge	0.0	0.0
11 :	education-chemistry	0.0	0.0
12 :	education-entrance exam	0.0	0.0
13 :	navigation-cab online	0.0	0.0
14 :	refrence-utility	8.6880976E-4	3.6924414E-4
15 :	medical-health	0.0	0.0
16 :	finance	0.0	0.0
17 :	multimedia-video	0.0	0.0
18 :	medical-baby	0.0	0.0
19 :	education-kids	0.0	0.0
20 :	business-job apps	0.007384883	0.0012163336
21 :	medical-diet	0.0	0.0
22 :	education-interview	0.0	0.0
23 :	navigation-maps	0.0	4.126846E-4

Fig 9. Confidence score calculate

The result based on the confidence score for both with and without LDA is shown through graph in Fig.11 and Fig. 12 respectively.

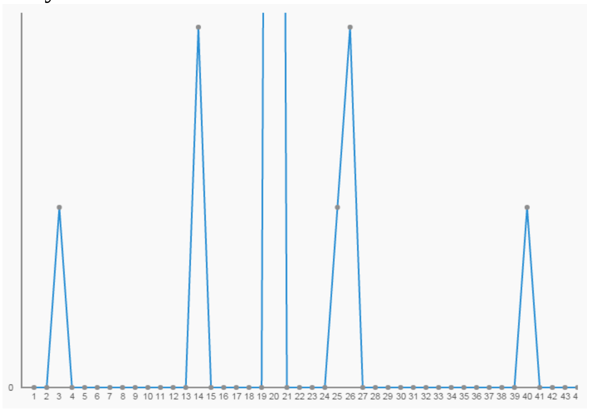


Fig.11 Results without LDA based on confidence score

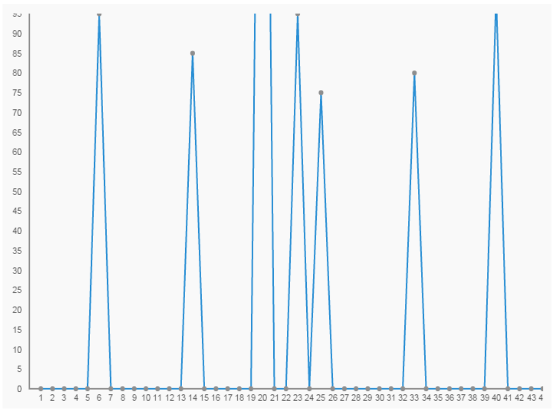


Fig.12 Results with LDA based on confidence score

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

VII. Conclusion

Effective classification of the mobile apps is important as everyday there are number of similar kind of apps coming in the market. Various classification techniques present for classifying the short and sparse data can be adapted for classifying the mobile apps. But the result obtained from these techniques does not give the effective classification of the apps. As they consider single factor for classification i.e. web knowledge or contextual information. So we have proposed an approach to effectively classify the mobile apps where we extract the information from multiple sources in order to improve classification that provides more effective result. We also considered leveraging the permissions requested by the apps which will then improve the ranking of the apps accordingly. Thus improving the security concerns of the malicious apps in easy to understand manner. Our system provides the user recommendations of the app based on his profile and also his preferences like apps with or without internet. This will not only help the user to select the proper app according to his requirements but also can be used for various other purposes like target advertising, user segmentation for market analysis etc.

References

1. H. Zhu,E. Chen,H. Xiong and H. Cao,``Mobile App Classification with Enriched Contextual Information," IEEE Transactions on mobile computing,Volume:13 , Issue: 07 ,7 July 2014
2. Christopher S. Gates, et.al,``Generating Summary Risk Scores For Mobile Applications",IEEE Transactions on dependable and secure computing, (Volume :11, Issue: 03), May-June 2014.
3. X.-H. Phan et al.,``A hidden topic-based framework toward building applications with short web documents," IEEE Trans. Knowl.Data Eng., vol. 23, no. 7, pp. 961–976, Jul. 2010
4. M. Sahami and T. D. Heilman,``A web-based kernel function for measuring the similarity of short text snippets," in Proc. WWW, Edinburgh, U.K., pp. 377–386,2006.
5. Z. Broder et al,``Robust classification of rare queries using web knowledge,"in Proc. SIGIR, Amsterdam, Netherlands,pp. 231–238,2007
6. H. Ma, H. Cao, Q. Yang, E. Chen, and J. Tian,``A habit mining approach for discovering similar mobile users," in Proc. WWW, Lyon, France, pp. 231–240,2012
7. H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian,``Exploiting enriched contextual information for mobile app classification,"in Proc. CIKM,Maui, HI, USA, pp. 16171621,2012.
8. W. Enck, P. Gilbert, B. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N Sheth,``Taint- Droid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, article 1-6,2010
9. A.P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner,``AndroidPermissionsDemystified,"Proc. 18th ACM Conf. Computer and Comm. Security, pp. 627-638, 2011.
10. E. Chin, A.P. Felt, V. Sekar, and D. Wagner,``Measuring User Confidence in Smartphone Security and Privacy," Proc. Eighth Symp. Usable Privacy and Security, (SOUPS '12), article 1, 2012.
11. B.P. Sarma et al.,``Android Permissions:A Perspective Combining Risks and Benefits," Proc. 17th ACM Symp. Access Control Models and Technologies (SACMAT 12), 2012.
12. William M. Darling,``A theoritical and practical implementaion Tutorial on Topic Modeling and Gibbs Sampling,"December 1, 2011
13. NaïveBayesClassification, http://www.saedsayad.com/naive_bayesian.html
14. K. Nigam et al.,``Using Maximum Entropy for Text Classification,"IJCAI Workshop Machine Learning for Information Filtering,1999,pp.61-67superscripts