Turkish Journal of Computer and Mathematics Education Vol.12 No.13(2021), 2235-2242

Research Article

Detection of False Ranking Apps Using Level Aggregation

R. Jeeva[a], N. Muthukumaran[b]

[a] Assistant Professor in Thamirabharani Engineering College
[b] Professor in Francis Xavier Engineering College
[a,b,] Tirunelveli, Tamilnadu,Affiliated to Anna University, Chennai
email:[a]jeeva3710@gmail.com, [b]kumaranece@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 4 June 2021

Abstract: Each platform of mobile devices has its app store which is the source for apps, games, movies, books, etc. The apps are categorized under predefined labels based on the rules formulated in the app store. The apps have been ranked based on the ratings, reviews, downloads, and no. of installs. It helps the user to download the top-ranked app in a specific category. That ranking of an app makes them think that it will work better than others in an effective way. The evidence aggregation of the above attributes has less variation that doesn't reflect the current status of an app which influences the ranking. For that, the attributes that have been frequently changed due to developer and user actions to be collected for a specific category in top charts. The attributes include version, last updated date, features of an app and keywords will undergo an independent process that produces the following levels: 1. Version change level, 2. Keyword matching level and 3. Feature matching level. Each value of a level has to be consolidated and aggregated to produce the final ranking of apps in a specified category. The actual ranking has been compared with the obtained ranking to find the deviation value and the false ranked app in the app store.

Keywords: Evidence aggregation, Version change level, Keyword matching level, Feature matching level

1. Introduction

In the app universe, ranking plays an important role to survive and sustain among a large number of applications. Many review platforms give a final picture of the user's expectation and the flow of user choice of apps with ranking and rating. As the Rating and Ranking decide the fate of the app's survivability, Google and Bing have their protective algorithms to rank the apps in top charts based on catchphrases and graphs. Users are incredulous of advanced showcasing, and companion proposals as application store evaluations and audits remain solitary as the single most prominent driver of revelation and transformation. Unmistakably application evaluations and surveys aren't simply vanity measurements; they have genuine results on change rates and brand notoriety. Most of the companies have a separate marketing section and put more effort both in terms of time and money for paid apps installations. The app developer has to pay a minimal amount to the marketing section or representatives for every installation and they boost the app's rating and ranking with increased visibility.

App Store and Play store have millions of apps in top trends, top grossing, and top paid with different categories and genres. But the app that exists in both the stores will not have the same rating or ranking. Of these applications, almost 66% have not gotten a solitary rating and 99% are viewed as unbeneficial. These examinations, in this manner, single out the uncommon exemptions for the standard—the best-ranked applications in each store.

Even though a large number of apps available in the market, the user's choice to download is still a difficult task. For that, App Store Optimization has been used where the managers obtain the no of downloads and the number of active users to increase the app's ranking and earn more money. A budget based on paid installation strategy has been formulated that boosts up the ranking lead to make the place in top trends or charts.

The rest of this paper is organized as follows. In Section 2, the evolution of app store optimization techniques has been discussed. Section 3 presents the architecture of the evidence aggregation system with different levels and its collection methods. Experiment results and analysis are shown in Section 4. The conclusions and further enhanced are given in Section 5.

2. Related work

Multihoming, a technique where a designer is distributing items for various platforms. Information assortment is in two stages. In the first stage, the scripts gather the novel identifiers of the applications which are generally well known in every category of the application store. In the second stage, the scripts gather the varied properties from the application's open sites at the application store. The assessment of multihoming application types was finished by breaking down their classifications in the application store. The level of accessible applications in every classification was determined and afterward contrasted with other application stores to spot the favored category alongside the mindful mobile application [1].

An integrated various levelled displaying approach has been utilized to quantify deals execution and affirm the outcomes with the utilization of the peril model and regression model. Application-level properties, for example, free application offers, high introductory positions, interest in less well-known categories, consistent quality updates, and high user survey scores impacts affect application manageability. Application-level investigation and vendor level examination have been performed with autonomous traits of an application that recognize the execution of the general deal of application sellers. It presumes that business execution was significantly higher for dealers while taking an interest in numerous classifications than something else. It didn't explicitly look at application explicit highlights or vendor explicit attributes that may influence the positioning of an application [2].

Messaging applications are one of the most well-known applications for mobiles with a great many dynamic users. The specification gave by generally well-known and business messaging applications has been examined. After recognizing the significant applications, light and old renditions have been sifted through. The essential highlights of an application have been recognized by displaying activities like send a message to an individual contact, read and reply to the message, include a contact, erase contact, and discussion. The optional highlights of an application must be distinguished like user profile, sending pictures, sound, and video. Keystroke level displaying has been tried by tallying the no of communications to accomplish the above essential and optional highlights. Utilizing the heuristic assessment of the above-inferred values, ease of use issues has been raised of texting applications [5].

China is one of the biggest android markets where the users can't get to the play store to purchase or introduce applications. For that, pre-introduced merchant explicit application markets and the number of autonomous application stores have developed. A review of an application incorporates gathering apk, different applications discharged by a similar designer, prescribed applications that are identified with it. At that point category, no. of downloads, min API level, outsider libraries included generally famous and publicizing categories with application ratings are gathered. In the wake of sifting through the cloned applications, hash estimation of apk has been checked and named single-store or multistore dependent on its accessibility [4].

3.Ranking Evidence system

From the app store like Google Play, appland, amazon apps, the information about every top ten application in all the categories is collected using the app bot. The information includes rating, review, and ranking and each parameter will undergo an independent evaluation which will be consolidated later. In rating, monthly data has been collected and calculate successive difference them. The results are checks against the threshold value that should be less than 1 set by the app bot.

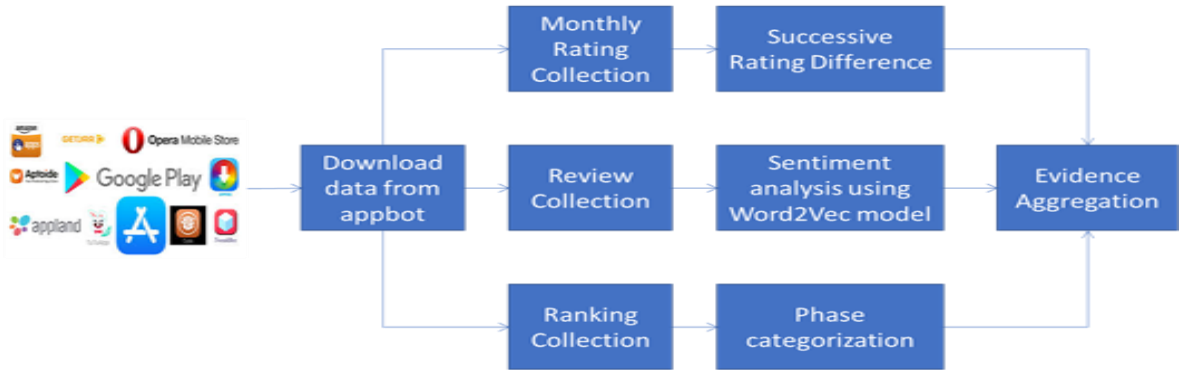

Fig 1. Aggregation of Ranking Evidence system

Those apps which have an increase in ratings more than the threshold value will take a negative value. In review, the collected data will undergo sentiment analysis using the Word2Vec model that predicts the target word based on the weight assigned to them. It evaluates reviews by splitting into tokens and undergoes stemming and stop word elimination processing. Then each word is compared with the keyword base for retrieving its weight. The weight of matched words with a keyword base is fused to form a review weight. In the ranking, categorize the data into the rising phase, maintenance phase, and recession phase to be passed for the next stage. In evidence aggregation, all the results of three parameters consolidated that determine the app's trustfulness in a store.

The rating, review, and ranking of an app will be gathered from the app store. Each app has its id that will be used to refer to an app store to retrieve the details like name, version, etc. The rating and ranking is a definite value that indicates the status of the app in the store. But the reviews can't be processed directly. The reviews of an app pass through the preprocessing stage that removes the stop words and performs data segmentation based on the collected words that are considered as labels. The sentiment analysis takes place to determine the nature of the

review which is positive or negative using the Word2Vec model. For that, the preprocessed data will undergo the extraction process which assigns a weight for the words matched with the key base. The words in the key base are updated after every successful completion process of sentiment analysis. The weight can be both positive and negative depending on the emotion of the word. The collective weight of a review will be checked against the average threshold. If it met the value, it will be considered as a positive otherwise it is negative.

The collected rank value will be different based on a category like top paid, top grossing, trending, etc. In each category, the rank value will be collected and the average of it will be considered as a final rank of an app. The difference between the rankings of successive categories has been marked and compared with the breaking limit. If it exceeds, the ranking will be marked for evidence evaluation. The same method has been followed for the monthly collected rating values to identify the deviation with the breaking limit. The total number of positives and negatives in the preprocessed reviews using sentiment analysis has been collected. The rating and ranking have been checked whether it has been marked. If the majority of the above three results in positive, then the app may consider as a genuine app. If not, the app will be marked as the fraudulent app that alerts the user and the app store.

4.Evidence Level Aggregation System

App Store has been playing the role of repository for various mobile apps with different categories at different levels based on specific parameters. There are 32 categories in the app store and each one has its ranking in ordering the apps based on rating, reviews, downloads, etc that can be collected.

Fig 2 shows that the character level matching of apps has to be collected and matched against the categorized ranked apps to identify the keyword matching level. The specific attributes like version, last updated are collected to identify how many the app has been revised after it has been launched which determines the version change level. The description of the app has been used to extract what are the primary features to be satisfied for the category that has been matched that determines the feature matching level. The levels are aggregated and compared with the category ranking to identify the deviation value. This value reveals how far or ahead of the apps have been ranked in the app store.

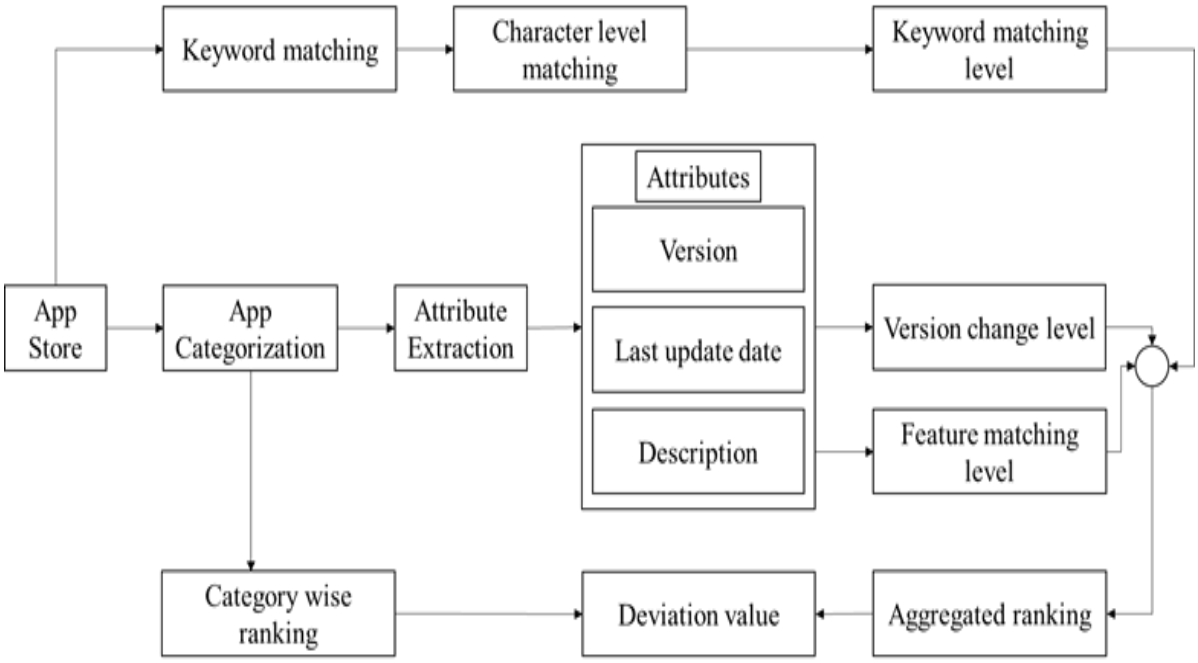

Fig 2. Architecture of proposed system

4.1.Keyword matching level

The keyword matching for the top-ranked apps in each category can be verified by exploring the characters from a to z. For each result, it shows 5 results which may be the listed in top categories. If a single character doesn't give the required results, increase the search text with the next character of the app name. It can be repeated with the remaining characters of the app name till a match occurs. If it matched, retrieve the no of characters and app name to determine the app search efficiency (ASV).

As shown in Fig 3, Search Impact Value (SIV) will be assigned with the varied ranges of ASV from 0 to 1. Placeholder Value (PV) will be calculated with the product of the app's place in the results and SIV. The average of ASV and PV can be expressed in percentile to determine the Keyword Matching Level (KML). If there is no match, the KML will be initialized to 0.

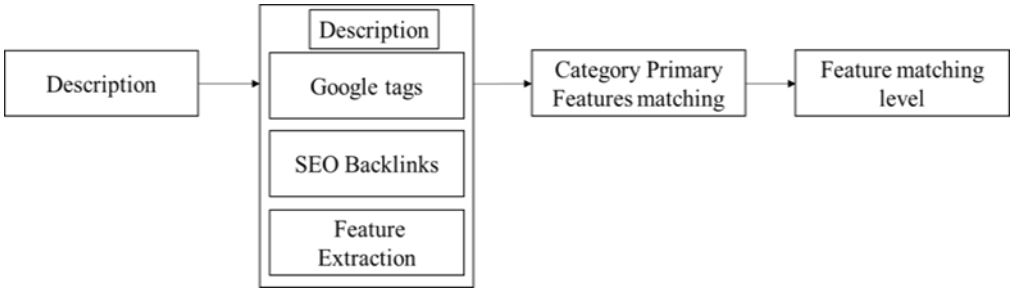

Fig 3. Keyword matching level

$$\text{ASV} = \frac{\text{No of chars required}}{\text{Length of app name}} \ldots.$$

$$\text{SIV} = \begin{cases} 1, & \text{ASV} \geq 0.4 \\ 0.2, & 0.39 \leq \text{ASV} \leq 0.3 \\ 0.6, & 0.29 \leq \text{ASV} \leq 0.1 \\ 0.8, & \text{ASV} < 0.1 \end{cases} \ldots.3$$

$$\text{PV} = \frac{5-\text{App place}}{5} * \text{SIV} \ldots.4$$

$$KML = \frac{ASV+PV}{2} \ldots.5$$

4.2.Version change level

A version of the app determines how many revisions and changes have been made that can be split into major and minor versions. Based on that, calculate how many versions have been released every year from the app launch that determines the average release of versions per year (ARVY). The difference between the last updated and its prequel will be calculated as an elapsed cycle (EC). The update frequency (UF) will be assigned to 1 if the value of EC is less than 12. If not, UF will be assigned as 0.

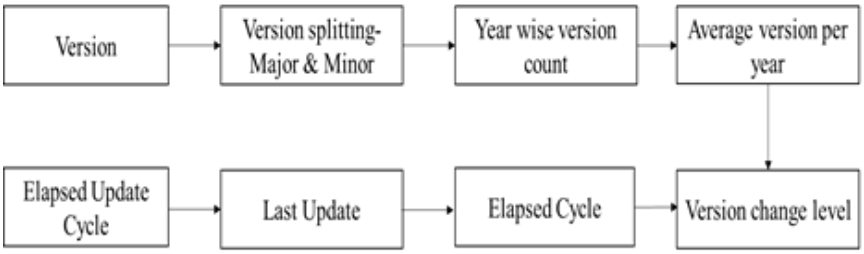

Fig 4. Version change level

$$UF = \begin{cases} 1, & EC < 12 \\ 0, & EC \geq 12 \end{cases} \ldots\ldots\ldots 6$$

$$VCL = \frac{ARVY*UF}{EC} \ldots\ldots 7$$

4.3.Feature matching level

Each category has its primary features to be satisfied irrespective of its additional features. After extracting the description of an app, identify the features F in the text using scrapping and store them in a repository. As the feature has not possessed a common name for all apps in the category, views V has been created for a different set

of features with unique weights. Each feature fi in F has been assigned a weight w that will be aggregated into a single view and match with the stored views to identify the optimal view for the app. The identified view weight has been averaged to the category feature score to determine the feature matching level (FML). Then identify how many SEO backlinks and Google tags have been used in the description excluding the duplicates to increase the app search optimization. Based on the matching occurrence in the repository of backlinks and tags, app optimization level (AOL) can be determined.

$F= \{f_1,f_2,\ldots f_n\}$

$W = \sum_{i=0}^{n} f_i(w)$

$$\text{FML} = \frac{\text{Weight of the identified view}}{\text{No of primary features of category}} \ldots..6$$

$$\text{AOL} = \frac{\text{No of identified backlinks and tags}}{\text{No of backlinks and tags in repository}} \ldots.7$$

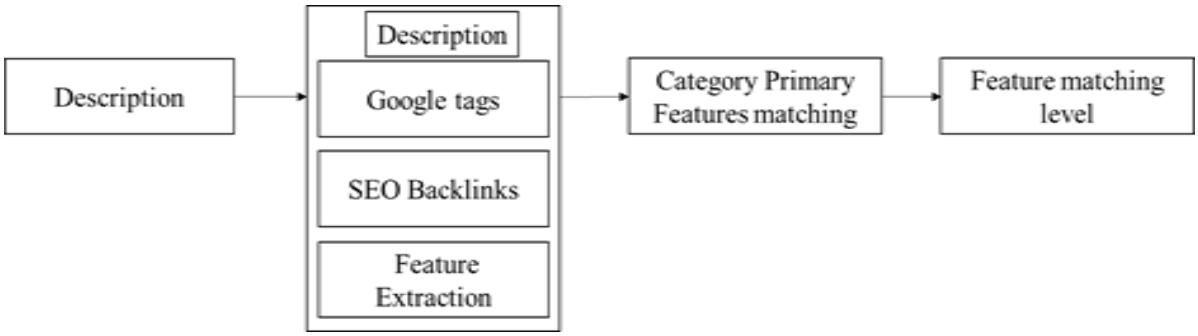

Fig 5. Feature matching level

4.4.Aggregated ranking

The identified levels are mapped to a scale of 1 to 5 with a range of values except for VCL as the convention is not the same. The product of computed scale that has been averaged with a consolidated value and VCL is termed as aggregated rank value. It will be compared with the collected category-wise ranking for the apps from the store to pinpoint the deviation of the rank.

5.Experimental Analysis

The app store data has been collected that holds 33 categories of 10,843 apps information like rating, installs, version, category, genre, last updated, android versions, etc. From that, the following attributes are chosen to identify the deviation levels. Table 1 shows the top-ranked apps rated by the concerned app store.

Table 1: Attributes of App

Rating	Size	Installs	Last Updated	Current Ver
4.1	19M	10,000+	7/1/2018	1.0.0
3.9	14M	500,000+	15/1/2018	2.0.0
4.7	8.7M	5,000,000+	1/8/2018	1.2.4
4.5	25M	50,000,000+	8/6/2018	Vary
4.3	2.8M	100,000+	20/6/2018	1.1
4.4	5.6M	50,000+	26/3/2017	1
3.8	19M	50,000+	26/4/2017	1.1
4.1	29M	1,000,000+	14/6/2018	6.1.61.1
4.4	33M	1,000,000+	20/9/2017	2.9.2
4.7	3.1M	10,000+	3/7/2018	2.8
4.4	28M	1,000,000+	27/10/2017	1.0.4

From that, the length of the app name and no. of chars required to search that app has to be calculated. The ratio of these two parameters gives the ASV. Using the equation, the SIV value has been calculated based on the

four ranges of values as shown in Table 2.

Table 2. Derivation of App search Efficiency and Search Impact Value

app length	No. of chars Required	ASV	SIV
46	5	0.108695	0.6
19	1	0.052631	0.8
52	23	0.442307	1
37	3	0.081081	0.8
26	11	0.423076	1
39	9	0.230769	0.6
16	2	0.125	0.6
20	10	0.5	1
29	9	0.310344	0.2
23	8	0.347826	0.2

Table 3. Keyword matching level for app places

app place	pv	KML
3	0.24	0.228696
3	0.24	0.330526
3	0.24	0.293077
1	0.8	0.859459
3	0.24	0.235385
4	0.04	0.404615
3	0.24	0.37
4	0.04	0.37
4	0.16	0.114483
4	0.16	0.123478

App place can be calculated with obtained SIV values that range between 1- 4. PV is calculated using the equation and the average of SIV and PV determines KML through the equation as shown in Table 3. The revision attribute reveals the number of major and minor versions of the app. The average version of an app (ARVY) is obtained with the multiplicative factors 0.5, 0.25, 0.125, and so on. The setting of UF can be obtained through the last updated date that has been compared with the current date fixed before the analysis. The VCL value is calculated with an equation with the derived value of UF and ARVY as shown in Table 4.

Table 4. Version change level with Elapsed Cycle.

ARVY	UF	EC	VCL
1	1	358	0.002793
2	1	350	0.005714
3	1	152	0.019737
1.5	1	206	0.007282
1	0	194	0.005155
1.5	1	645	0.002326
21.875	1	249	0.087851
7	1	200	0.035
6	1	467	0.012848
2	0	181	0.01105

Based on the category, the primary features required to be in the app are different. The primary features F for each category have been identified that match with the available features in the app to determine feature weight

FW. The ratio of FW and F determines the FML as specified in the equation. Then the ratio of the available backlinks for each app has been collected from app store data to the required determines AOL as specified in the equation in Table 5.

Table 5. App Optimization level with backlinks.

AF	FW	FML	BL NO	AOL
6	12	0.857142857	19	0.633333333
4	8	0.571428571	23	0.766666667
6	12	0.857142857	14	0.466666667
4	8	0.571428571	12	0.4
5	10	0.714285714	22	0.733333333
6	12	0.857142857	9	0.3
5	10	0.714285714	9	0.3
5	10	0.714285714	4	0.133333333
4	8	0.571428571	6	0.2
7	14	1	16	0.533333333

The obtained KML, VCL, FML and AOL are aggregated to determine the new ranking efficiency (ORE) that has been compared with the actual (ARE) for identifying the deviation value as shown in Table 6.

Table 6. Obtained and Actual App Efficiency based on Level aggregation

KML	VCL	FML	AOL	Agg	ORE	ARE
0.63	0.0028	0.57	0.47	0.42	40	75
0.38	0.0057	0.71	0.27	0.34	65	85
0.14	0.0197	0.57	0.3	0.26	95	5
0.25	0.0073	0.86	0.83	0.49	50	25
0.46	0.0052	0.57	0.27	0.33	20	60
0.16	0	0.57	0.47	0.3	45	35
0.4	0.0879	0.71	0.43	0.41	5	90
0.14	0.035	0.71	0.37	0.31	30	75
0.16	0	0.86	0.77	0.45	70	35
0.65	0.0111	0.86	0.5	0.5	80	5

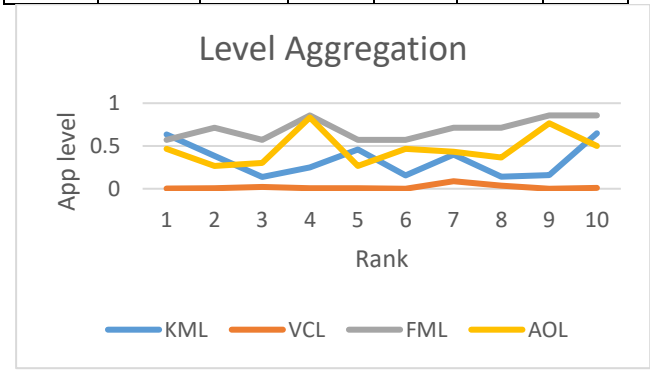

Fig 6. Mapping of KML, VCL, FML and AOL

Fig 7. Rank deviation based on level aggregation

6.Conclusion

In this paper, the values from the app's attributes have been used to derive the VCL, FML, KML, and AOL. As these values are dynamic, the interval has been set for the data collection and updating. After the level aggregation, the rank deviation has been shown that has discrepancies from 30 to 90% and the app that has higher values in all the levels results in better ranking. The app's ranking is not only based on the user side. From the developer's view parameters also, the app ranking can be influenced for better results.

References

1. Sami Hyrynsalmi, Tuomas Makil, Antero Jarvi, Arho Suominen, Marko Seppanen and Timo Knuutila.: App Store, Marketplace, Play! An Analysis of Multi-Homing in Mobile Software Ecosystems. Proceedings of IWSECO 2012,59-72.
2. Gunwoong Lee and T. S. Raghu, Determinants of Mobile Apps Success: Evidence from App Store Market, Journal of Management Information Systems,Volume 31, 2014 - Issue 2, 1-46.
3. Caro-Alvaro, Sergio & García, Eva & García-Cabot, Antonio & de-Marcos, Luis & Gutiérrez-Martinez, Jose-Maria. (2017). A Systematic Evaluation of Mobile Applications for Instant Messaging on iOS Devices. Mobile Information Systems. 2017. 1-17. 10.1155/2017/1294193.
4. Wang, Haoyu & Liu, Zhe & Liang, Jingyue & Vallina-Rodriguez, Narseo & Guo, Yao & Li, Li & Tapiador, Juan & Cao, Jingcun & Xu, Guoai. (2018). Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets.
5. Jeeva. R, Muthukumaran. N, 'Identification of Fictitious Messages in Social Network using E-Hits and Newsapi', International Journal of Innovative Technology and Exploring Engineering, Vol. 9, Issue. 10, pp. 7- 11, August 2020.
6. Karthika. A, Muthukumaran. N, Joshua Samuel Raj. R, 'An Ads-Csab Approach for Economic Denial of Sustainability Attacks in Cloud Storage', International Journal of Scientific & Technology Research, Vol. 9, Issue. 04, pp. 2575-2578, April 2020.
7. Zahra, Fatima & Hussain, Azham & Haslina, Haslina. (2017). Usability evaluation of mobile applications; where do we stand?. AIP Conference Proceedings. 1891. 020056. 10.1063/1.5005389.
8. J. Rebekah, D. C. J. W. Wise, D. Bhavani, P. Agatha Regina and N. Muthukumaran, "Dress code Surveillance Using Deep learning," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 394-397, doi: 10.1109/ICESC48915.2020.9155668.
9. Chen, Ying & Xu, Heng & Zhou, Yilu & Zhu, Sencun. (2013). Is this app safe for children?: a comparison study of maturity ratings on Android and iOS applications. 201-212. 10.1145/2488388.2488407.
10. Zhu, Hengshu & Xiong, Hui & Ge, Yong & Chen, Enhong. (2015). Discovery of Ranking Fraud for Mobile Apps. IEEE Transactions on Knowledge and Data Engineering. 27. 74-87. 10.1109/TKDE.2014.2320733.
11. Hu, Bing & Liu, Bin & Gong, Neil & Kong, Deguang & Jin, Hongxia. (2015). Protecting Your Children from Inappropriate Content in Mobile Apps. 1111-1120. 10.1145/2806416.2806579.