A Recommendation System for Emergency Mobile Applications using Context Attributes: REMAC

Alireza Ahmadi
alireza.ahmadi1@ucalgary.ca
SEDS laboratory
University of Calgary
Calgary, Canada

Debjyoti Mukherjee
debjyoti.mukherje1@ucalgary.ca
SEDS laboratory
University of Calgary
Calgary, Canada

Guenther Ruhe
ruhe@ucalgary.ca
SEDS laboratory
University of Calgary
Calgary, Canada

ABSTRACT

The extensive use of mobile devices had led to tremendous growth in not only the usage of different apps but also their capability to help people in moments of crisis. There are different emergency mobile apps published in the app markets; these apps can be of enormous assistance to victims as they can provide valuable information and guidance at the opportune moments. However, app store reviews, ratings, and relevant studies have revealed that users are often averse to using these apps or their different features. This draws our attention to the need for recognizing essential features and including them in the emergency apps to increase their usability. Our proposed recommendation system called REMAC combines different machine learning techniques to analyze the context characteristics of different organizations and suggest unique features that can be included in their emergency apps. REMAC is built by analyzing 24 potential context attributes of 1909 universities spread across North America. This research also includes a systematic attribute selection process that enables us to reach a local optimum for the given dataset. This tool carefully dissects the context attributes of each university and suggests top features that should be included in its emergency app. It leverages the data (other apps and features) provided by the app markets to suggest essential features.

Even though emergency apps can be in different types and categories, in this study, universities' emergency apps have been investigated which have similar nature. REMAC is evaluated by inspecting features from 41 apps, and the tool has an accuracy of over 97.07%. In the course of this research, we found that (i) emergency apps from organizations having similar context attributes share a high degree of overlapping features (ii) top common features of emergency apps in entities of one cluster can be a good recommendation for the app of another entity similar to them, and (iii) for a given dataset, a subset of the data can also be used to enhance the performance of the system without compromising on the accuracy. REMAC has been parameterized to adjust according to the requirements of the organizations. It can proactively understand organizations' needs and suggest features to support their emergency apps. REMAC can be further enhanced to look beyond the current set of functionalities provided by emergency apps and extract other utilitarian features by assessing users' needs.

WAMA '19, August 27, 2019, Tallinn, Estonia

ACM ISBN 978-1-4503-6858-2/19/08…$15.00
https://doi.org/10.1145/3340496.3342760

CCS CONCEPTS

• **Software and its engineering → Requirements analysis**; *Software prototyping*.

KEYWORDS

Emergency, Mobile Apps, Recommendation System, Context, Clustering

ACM Reference Format:
Alireza Ahmadi, Debjyoti Mukherjee, and Guenther Ruhe. 2019. A Recommendation System for Emergency Mobile Applications using Context Attributes: REMAC. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics (WAMA '19), August 27, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3340496.3342760

1 INTRODUCTION

Smartphones can be of enormous importance in emergency situations, as they may be among their owner's only possessions and resources [12]. To mitigate the threats of an emergency situation and being cognizant of the importance of mobile functionality during any crisis, several mobile apps have been designed by NGOs (non-government organizations), official state departments, international humanitarian organizations, and other individual companies. The goal of all these different apps is to facilitate the management of crisis and provide guidance to the impacted distressed victims. However, it has been identified that apps currently available to the general public lack relevant and vital features [20].

However, including all the important features for various types of emergencies would lead to the problem of too many features being included. This is not an ideal situation as the victims will be unnecessarily puzzled navigating through the myriad of options to find the desired features. Using a complex system in the middle of any crisis can be more stressful and can consume a lot of time. Since time is critical in an emergency situation and even saving a few seconds can play a vital role in a victim surviving a crisis, it is of paramount importance that emergency apps should have features that are most relevant to the current situation [30].

For any mobile emergency app, this can be made possible if context information is available and the features are prioritized using contextual information[18]. Contextual information refers to all the factors that can potentially play a role in an emergency

situation. We have built a recommendation system (REMAC[1]) that can propose vital features to be included in the emergency app using contextual information. For this study, we have focused only on university emergency apps, but the same approach can be used for other cases as well. In this research, we have demonstrated how contextual information can be effectively used to categorize different universities and then suggest top features based on their category. This recommendation system clusters similar universities based on university attributes and then recommends top features that are common for universities in that cluster. When a new university data is obtained, it can be mapped to one of the existing clusters using its context attributes and the proposed features can be included in the new university's emergency app. In this paper, "apps" refers to mobile emergency apps only.

In this paper, we answer the following research questions:

RQ1: How good can the contextual information be used in a recommendation system to cluster similar entities together?
Why and How? The nature and impact of any emergency situation depend on the characteristics of its context [23]. Identifying these characteristics and their influence on emergency situations will enable us to build an effective recommendation system. We will extract relevant context characteristics data and cluster similar ones. We will use existing features of safety apps in our dataset to assess the similarity between them.

RQ2: Can we suggest useful emergency mobile apps' features for an entity based on its context similarities to other entities?
Why and How? Considering the concept of collaborative filtering, it is highly likely that entities with similarities could have similar requirements for their safety mobile apps. We plan to build a system that would gather information and suggest features using it. We will evaluate the accuracy of our tool for recommending features using existing features of safety apps in our dataset.

We have organized this paper in the following manner. Section 2 describes the previous work done in this area, section 3 refers to the dataset, and section 4 describes the methodology that we have used. Section 5 contains the result, sections 6 refers to the threats to validity, and section 7 contains the conclusion and the future work.

2 RELATED WORKS

Recommendation systems leverage an input data in a way to generate desired output. Therefore, recommendation systems differ based on "what to consider as input", "how to process it", and "what to recommend as output" [25]. Considering the nature of recommendation systems and the field of study, we can divide the related studies into two main categories; the recommendation systems which use contextual information and the ones which are used for eliciting requirements. In this section, we are going to mention some of these studies.

Context-Based Recommendation Systems: Context sensitive systems are ubiquitous and there is a lot of research in this field. The focus of all these studies is to make the systems more effective so that it can adapt to its surroundings. In [4], Matthias conducted a survey on various context-aware systems to present common architecture principles of context-sensitive systems and also derive a design framework to explain the various elements common to most context-aware architectures. In another study [28], Vaninha et al. presented an integrated approach for designing context-sensitive systems (CSS). This study focused on the proposed way of thinking about context, on the proposed context metamodel and on the specification of a process for designing CSS. Gediminas Adomavicius and Alexander Tuzhilin conducted a study in [2] to assess the importance of contextual information for recommendation systems. They argued that relevant contextual information does matter in recommender systems and that it is important to take this information into account when providing recommendations. They further discussed how context can be modeled in recommender systems. In another study [27] conducted by Katrien et al., they conducted a survey on context-aware recommendation systems for learning. As learning is taking place in extremely diverse and rich environments, the incorporation of contextual information about the user in the recommendation process has gained a lot of interest. Such contextualization is researched as a paradigm for building intelligent systems that can better predict and anticipate the needs of users, and act more efficiently in response to their behavior. A systematic literature review on context-aware recommendation systems [29] was performed by Norha et al.. All these studies and many more have emphasized how contextual information can be effectively utilized in different software systems. There are several other studies [3, 16, 22] discussed how different contextual information has enhanced the functionality of the systems. But none of these studies have explicitly worked with mobile emergency apps or discussed how contextual information can be used for eliciting vital features for emergency apps.

Recommendation Systems for Software Feature: Several studies have been conducted to elicit the most essential and important requirements for software. Most of these studies leverage the role, importance, and interests of stakeholders as input to a system in order to recommend the right features. In [8], a recommender system for requirement elicitation is introduced which uses unsupervised clustering techniques to manage the placement of stakeholders and their interests for recommending features. In some other studies, by using data mining, the same concern for ultra large scale software has been addressed [10][7]. In [17], Lim and Finkelstein proposed a method for identifying and prioritizing requirements by using social networks and collaborative filtering. Even though in some other studies such as [19] and [21], different types of inputs like users' feedback, value and effort of the features, competitors' useful features, and cohesiveness between features have been investigated, a recommender system which automates the process has not been proposed.

To the best of our knowledge, this study is the first which considers using contextual information for recommending features to an emergency mobile app.

[1] **R**ecommendation system for **E**mergency **M**obile **A**pplications using **C**ontext

3 DATASET

In this study, we have used two sets of data from two different sources. In this part, we are going to describe the process of data collection and the structure of the data.

3.1 Mobile Emergency Apps' Features

As mentioned earlier, our focus is on emergency mobile apps of the universities. In order to gain valuable insights in this area, we have collaborated with AppArmor, a company pioneer in developing custom mobile safety apps and having over 200 customers around the world, mainly in North America. Most of these customers are educational institutions including universities. Therefore, in this study, emergency apps of the universities in North America have been targeted. AppArmor provided us the *list of the apps* developed by them along with the features.

In this research, we have focused on universities. So, we retrieved only those apps from this data that belonged to universities. Our final list contained apps from 41 different universities. We extracted the distinct features present in each app; the average number of features per app is 8, and the total number of unique features across all the apps is 19.

3.2 Universities' Context Attributes

We analyzed different websites and datasets to extract university context related attributes. We found *UniRank* website as one of the sources which provide the most relevant information about different context aspects of the universities [1]. For example, UniRank is the only source which includes the settings of the universities' campuses; this can be a vital piece of contextual information. We developed an automated web scraping tool to scrap this website and extract the information for all the universities in North America.

We successfully collected data related to 24 different university attributes for 1909 universities across three countries of North America: Canada, United States, and Bermuda. Among these universities, we selected only those 41 that were common to AppArmor's customer list. The university attributes that we collected were: country rank, founded year, country, state or province, city, city population range, graduate studies, undergraduate studies, gender admission, admission selection, admission rate, number of students, number of academic staff, control type, entity type, academic calendar, campus setting, religious affiliation, library, housing, sport facilities, financial aids, abroad study, and distance learning.

4 METHODOLOGY

In this section, we discuss the main concepts of REMAC. Figure 1 describes the process flow of the system.

4.1 Context Attributes Selection

As discussed in Section 3, we scrapped 24 attributes as context-related university attributes. On further scrutinizing, we realized that all the attributes may not be essential to achieve the best results. We used a systematic attribute selection mechanism to obtain the optimum set of attributes following these steps:

- We removed the attributes that had the same value for all 41 universities since they are not going to affect REMAC. For example, as all the universities have a library, we removed this attribute.
- We calculated the variance for the numerical and the categorical binary attributes. We eliminated those attributes whose variance was less than a threshold. For example, in some attributes such as the calendar type of universities or the presence of sports facilities in them, only a very few universities were different from the others. So, we removed these attributes.
- For the categorical attributes with the multiple values, we calculated the frequency for each value. We determined the best and the worst distribution for each attribute and computed the distance of the current distribution from these two extremes. We eliminated those attributes which were closer to the *worst distribution* based on a threshold. Following this step, attributes like religious affiliation have been removed, as there is "no religious affiliation" for most of the universities.

Generally, we kept attributes which may provide a distinction between universities. Applying this process on the entire dataset, we eliminated half of the attributes; then we followed a greedy approach to check if the accuracy of REMAC can be enhanced by including some of the discarded attributes. We checked with the different combinations of attributes and obtained the local optimal set which also contained 12 attributes. From now, we name the set of attributes in the local optimum as the "*direct attributes*". We will evaluate the efficiency of the REMAC on both sets of all 24 attributes and 12 direct attributes.

4.2 Clustering

Clustering is the task of grouping a set of elements in such a way that elements in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters) [5]. REMAC uses clustering techniques to group the universities into different clusters based on their context attributes. The university attributes that we have selected have a mixed datatype; some of them are numerical data (for example, City population, Admission rate, Academic staff, etc.), whereas the others are categorical types (for example, City, Region, Country, Admission selection, etc.). Of the 24 attributes, only 8 are of a numerical type and the remaining 16 are of a categorical type. So we needed a clustering algorithm that could work with such a mixed datatype and K-Prototype was the ideal option for us [13].

K-Prototype is a combination of 2 famous clustering algorithms, namely K-Means [5] and K-Modes [9].K-Means only works with numerical data while the K-Modes clustering algorithm is an extension to the K-Means algorithm for clustering categorical data. K-Prototype works by combining both these 2 algorithms together; it applies K-Means for the numerical attributes using Euclidian distance as the measure for calculating distance, and K-Modes for the categorical attributes using dissimilarity measure as the distance function [13].

K-Prototype has two main parameters that need to be adjusted to obtain the best clustering results named "K" and "γ". K refers to the number of clusters into which the entire data has to be divided. In order to pick K wisely, we used the "Elbow Technique"[14]. γ

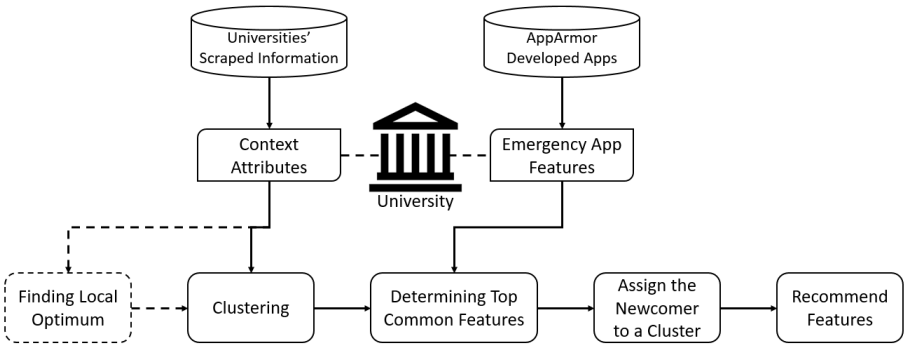

Figure 1: Our process

is the weight assigned to the categorical attributes with respect to the numerical ones. The optimum value for γ was decided based on the highest similarity score (discussed in section 4.4).

4.3 Recommendation System

In this study, we have developed a system using the concept of collaborative filtering. Collaborative filtering is a method used by recommendation systems, which by collecting preferences and information from many entities, makes automatic predictions for another entity [6]. The underlying assumption of the collaborative filtering approach is that if two entities are similar in one aspect, they are more likely to be similar in another aspect too [26].

In our case, we use clustering analysis to group similar universities into clusters. We determine the top common features that are present in the apps of the universities in each cluster by calculating the frequency of features' occurrences. We believe that these features are good candidates to be implemented in the app of a new university that has similar attributes. In other words, we are using the attributes of a university as a knowledge to recommend useful app features to it. We judge the effectiveness of REMAC by calculating the "Accuracy" of the system as described in 4.4.

4.4 Validation

There are two main steps in REMAC:

(1) Making clusters based on *context attributes* to have similar universities in one set.
(2) Recommending features based on *common features of the apps* from the universities within the same cluster to a new university similar to them.

In order to validate our tool, we need to assess the quality of the clusters and then measure the usefulness of the recommended features. In the following, we suggest two similarity scores for clusters' quality assessment and then discuss the customized version of k-fold cross validation for evaluating the recommendation part of the system. In the end, we express the definition of the term "accuracy" for the REMAC

Similarity Score: For measuring the quality of the clusters, we propose a criteria named *"similarity score"* using *Jaccard Index* and *Cosine Similarity*.

If A is the set of features available in app of university U_A and B is the set of features available in app of university U_B, we can calculate the similarity between U_A and U_B using Jaccard index as denoted in equation 1 [24].

$$JaccardIndex = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

We map each set of app features to a binary vector with a fixed length of m, where the m is the total number of features that can potentially be available in an app. As mentioned in section 3, based on our dataset, $m = 19$. We build the feature bit vector such that each element in the vector is either "1" (if the corresponding feature is implemented in the app), or "0" otherwise. If A is the mapped vector from the set of features available in safety app of university U_A and B is the mapped vector from the set of features available in safety app of university U_B, we calculate the similarity between U_A and U_B using Cosine similarity as denoted in equation 2 [31].

$$CosineSimilarity = \frac{A.B}{|A||B|} \tag{2}$$

We calculated similarity within a cluster by obtaining the average of Jaccard index (or Cosine similarity) values of app features for each pair of universities in the cluster. Then we obtained the *similarity score* by calculating the average of the similarity values for all the clusters. We named it *Similarity Score 1* when we used Jaccard index and *Similarity Score 2* when we used Cosine similarity. It is obvious, that for a better comparison, we need to know the similarity score of the set before clustering. In that case, we assumed that all the universities are in one cluster and followed the same process. Following this process, we can assess the quality of clusters using a criterion dependent from clustering algorithm; while the clustering process has been done based on context attributes and similarity scores have been calculated using apps' features.

K-Fold Cross Validation: Cross validation is a validation technique for assessing the performance of supervised learning methods on labeled data [15]. In our case, we used this technique to assess the performance of the clustering method, where there are no labels. We divided our data into k folds and perform the clustering (finding centers) using (k-1) folds of data. We calculate the similarity scores based on the (k-1) folds of data and also determine the top

common features for each cluster. Suppose we have obtained n clusters $C_1, C_2, ..., C_n$. We mapped each university U in the remaining fold (referred to as newcomer) to one of the clusters by calculating its distance from the n cluster centers and selecting the minimum one. In case of multiple minimal distances, alphabetical ordering is applied. We recalculated the similarity scores on the entire data and iterated this procedure for k times. We obtained the "Accuracy" measure as discussed below.

Accuracy: Accuracy is the measure of the preciseness of the suggested features using REMAC. As discussed above, we used k-fold cross validation to evaluate REMAC. Suppose a newcomer university u is mapped to a particular cluster C_i, for which the set of top common features is $F = \{f_1, f_2, ..., f_l\}$. We calculate the accuracy by measuring the overlap between F and the features of the app for u. The number of top common features has been set to 5 as default. We call an overlap between a newcomer app features and a set of top common features as a *good and acceptable overlap* if the number of overlaps is 3 or more. While increasing these numbers (e.g. the overlap of 5 and more out of the top 7 features) can raise the chance of high accuracy, we decided to stick to the overlap of 3 and more out of the top 5 features in order to stay strict enough.

5 RESULTS

In the previous section, we explained different modules of REMAC. Now, we are going to show that REMAC makes meaningful clusters and recommends useful features. In this section, we have formulated three hypotheses based on an assumption and then evaluated them.

5.1 Assumption and Conjectures

Assumption: *Requirement engineers of AppArmor extract the similarities of a new client with the previous ones and suggest the same features to them.*

First Conjecture: *Universities with similar context attributes have similar features in their safety apps.*

Based on this assumption, we expect to see universities' apps in each cluster based on the context attributes are more similar. For calculating similarity, we would use the methods mentioned in section 4.4.

Second Conjecture: *Top common features of apps from universities in one cluster could be good recommendations for the safety app of another university similar to them.*

Our goal is to recommend features for safety mobile apps. We expect to see REMAC is recommending useful features for a new university.

Third Conjecture: *Considering direct attributes of context would result in more useful features for universities' apps.*

In section 4.1, we decreased the number of context attributes following a systematic procedure and we named the new set as direct attributes. We expect to see that using this set would increase the accuracy of REMAC in recommending features.

5.2 Evaluation of Conjectures

In order to evaluate conjecture 1, we ran K-prototype 5 times on the set of 41 universities. We obtained the optimum number of clusters ("K") as 4 using the Elbow technique. At each run, we calculated the similarity score of the set after clustering and compared the results to the similarity score of the set before clustering. The results are reported in table 1. As shown, both the similarity scores have always increased after clustering. We can say that REMAC produces meaningful clusters. Therefore, we have no reason to reject the first conjecture.

In order to evaluate the second conjecture, we calculated the accuracy of REMAC considering 5 top common features and measuring the overlap of 3 or more features. We ran the 10-fold cross validation for 10 times and obtained the accuracy of the tool as 97.07%. The frequency of the overlap counts is shown in figure 4. As evident from the experiment, there is no university safety app with features overlaps count of 0 or 1 and there is only one university with overlaps count of 2 features. The result of similarity scores are also shown in figures 2 and 3. As expected, in all the folds, there is a considerable increase in similarity score after clustering to that before clustering. Only in 3 occasions, the similarity score after clustering newcomers has been less than after clustering the base set only. These decreases are due to additional features of newcomers which will not affect the efficiency of REMAC. Therefore, we have no reason to reject the second conjecture.

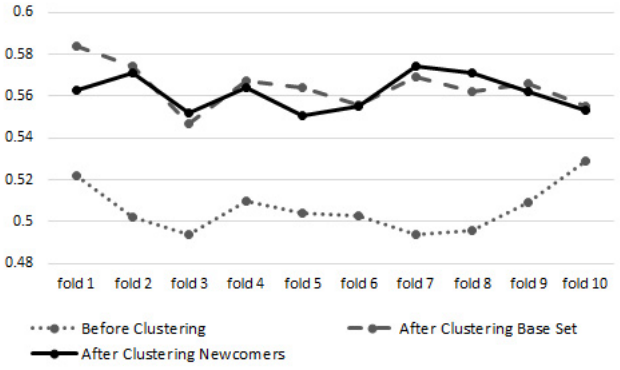

Figure 2: Similarity Score 1, before and after clustering on 10 folds

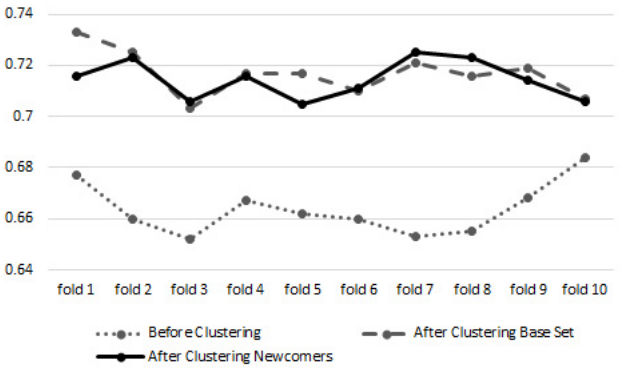

Figure 3: Similarity Score 2, before and after clustering on 10 folds

Table 1: Achieved similarity scores from the set of 41 universities before and after clustering over 5 runs

	Similarity Score 1 Before Clustering	Similarity Score 2 Before Clustering	Similarity Score 1 After Clustering	Similarity Score 2 After Clustering
Run #1	0.506	0.664	0.543	0.697
Run #2			0.561	0.713
Run #3			0.541	0.696
Run #4			0.564	0.716
Run #5			0.563	0.716

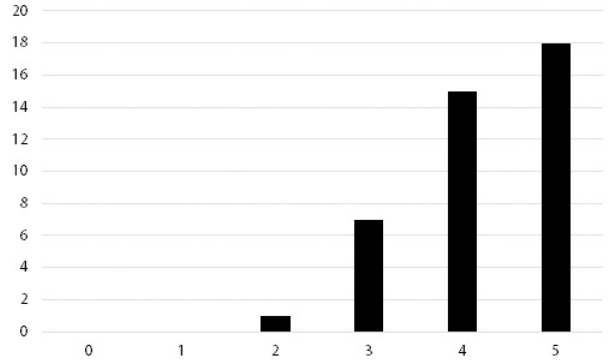

Figure 4: Features' overlap count distribution

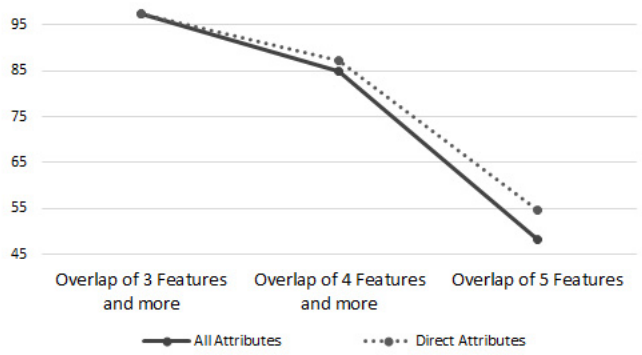

Figure 5: Average of accuracy in different levels over 5 runs

For the third conjecture, we evaluated REMAC using two scenarios. First, considering all the 24 attributes and then, considering only the 12 direct attributes.

When we calculated the accuracy considering the overlap of 3 or more features out of 5, the results were the same. However, as we make the conditions more stringent, and calculate accuracy considering only overlap of 4 or more, or all 5 features, the value varies. As evident from figure 5, the accuracy of REMAC has improved using direct attributes. These values have been obtained by calculating the accuracy over 5 runs. Therefore, we have no reason to reject the third conjecture as well.

6 THREATS TO VALIDITY

We reported the results of REMAC for emergency mobile apps (based on the similarity measures for the clusters and the feature overlap of the recommended features) to describe the process and demonstrate its effectiveness of the system. In the following, we discuss the threats to validity [11].

Construct Validity: REMAC is based on the assumption about how AppArmor requirement engineers gather information and suggest app features to their clients. We also evaluate the system based on the feature overlap for the emergency apps of their clients. However, considering the huge client base that AppArmor has and knowing that they are a pioneer in developing custom emergency apps, we trust their knowledge and the capability of suggesting important features based on university characteristics. Our system stands on two pillars; context attributes for universities and features suggested by AppArmor for their apps. There is no doubting the fact that features provided in the university's emergency apps are highly dependent on their different context criteria. In building this system, we have chosen those context attributes that can be influential in emergency cases.

Conclusion Validity: REMAC had access to features of 41 university emergency apps and the apps had diverse features. It has been seen that increasing the similarity score over 19 potential features needs a high degree of overlap. In all the experiments, it has been confirmed that clustering has provided better similarity scores. The accuracy of REMAC for suggesting top features has been calculated on 10 runs using k-fold cross validation. Although this result may vary for a different dataset, the results will still be comparable.

Internal Validity: Since REMAC works by clustering similar data points based on their context attributes, it is vital to choose context attributes appropriately. Inevitably, the results would suffer if these attributes are selected in an ad-hoc manner without due consideration. In the case of REMAC, we picked related context attributes by considering different sources of data. Furthermore, we optimized the system by finding the local optimum of the dataset.

External Validity: We have evaluated REMAC using published emergency apps built by AppArmor. Owing to the unavailability of other data sources, our evaluation is limited to this data set. Hence, we cannot claim the general applicability of this system on other data. This system stands on the fundamental of achieving good clusters based on appropriate context attributes, and that does not vary with data. Generalizing this approach is highly dependent on the context attributes. We believe REMAC should work on other data as long as there are valid context attributes and efficient evaluation data sets.

7 CONCLUSION AND FUTURE WORK

In this study, we developed a recommendation system that would suggest important features to a new university's emergency app. We evaluated this tool by running 10- fold cross validation for 10 times and obtained an average accuracy of 97.07% when we considered the overlap of 3 or more features among the top 5 features suggested. The key idea is to group similar universities and then extract the common features from them. These common features can be suggested to any new university that is added to the cluster.

As more and more universities are added to the clusters, the feature set would gradually change and so would be the suggested features. So, the clustering algorithm should be run frequently as other universities are added to the list. As we discussed in the paper above, we started with an assumption and three hypotheses. We discovered that all of the hypotheses were true. The results are based on the dataset that was available during this experiment. With a different dataset, the results might vary.

Considering the process flow of the REMAC, there are some areas that can potentially be investigated in the near future. We have listed those areas below:

- **Applying REMAC to other datasets:** Till now, we have built and evaluated REMAC in emergency apps of universities, while we believe this approach can be used for other applications as well. We will use other datasets to assess the efficiency of the REMAC and discover the potential changes needed in it.
- **Looking for approaches for finding context attributes and their relevance:** Context attributes chosen for universities resulted in high similarity in the clusters and high accuracy in recommending features. As we mentioned, using REMAC is highly dependent on picking right context attributes and this can be a challenging step for other contexts. We would like to investigate the process of picking attributes and measuring the relevancy of them.

ACKNOWLEDGMENTS

We thank AppArmor for providing valuable insights and helpful inputs to this study. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC
Discovery Grant RGPIN-2017-03948.

REFERENCES

[1] Accessed: 2019-03-09. World University Rankings & Reviews | uniRank. https://www.4icu.org/.
[2] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender systems handbook*. Springer, 217–253.
[3] Sabah Al-Fedaghi. 2017. Context-aware software systems: toward a diagrammatic modeling foundation. *Journal of Theoretical and Applied Information Technology* 95, 4 (2017), 936.
[4] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. 2007. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* 2, 4 (2007), 263–277.
[5] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 59–68.
[6] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 43–52.
[7] Carlos Castro-Herrera, Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. 2008. Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 165–168.
[8] Carlos Castro-Herrera, Chuan Duan, Jane Cleland-Huang, and Bamshad Mobasher. 2009. A recommender system for requirements elicitation in large-scale software projects. In *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 1419–1426.
[9] Anil Chaturvedi, Paul E Green, and J Douglas Caroll. 2001. K-modes clustering. *Journal of classification* 18, 1 (2001), 35–55.
[10] Jane Cleland-Huang and Bamshad Mobasher. 2008. Using data mining and recommender systems to scale up the requirements process. In *Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems*. ACM, 3–6.
[11] Robert Feldt and Ana Magazinius. 2010. Validity threats in empirical software engineering research-an initial survey.. In *Seke*. 374–379.
[12] Janey Gordon. 2007. The mobile phone and the public sphere: Mobile phone usage in three critical situations. *Convergence* 13, 3 (2007), 307–319.
[13] Zhexue Huang. 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery* 2, 3 (1998), 283–304.
[14] Trupti M Kodinariya and Prashant R Makwana. 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal* 1, 6 (2013), 90–95.
[15] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Montreal, Canada, 1137–1145.
[16] Florian Künzler, Jan-Niklas Kramer, and Tobias Kowatsch. 2017. Efficacy of mobile context-aware notification management systems: A systematic literature review and meta-analysis. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 131–138.
[17] Soo Ling Lim and Anthony Finkelstein. 2011. StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. *IEEE transactions on software engineering* 38, 3 (2011), 707–735.
[18] Faisal Luqman and Martin Griss. 2010. Overseer: a mobile context-aware collaboration and task management system for disaster response. In *2010 Eighth International Conference on Creating, Connecting and Collaborating through Computing*. IEEE, 76–82.
[19] Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. 2009. When users become collaborators: towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, 981–990.
[20] Maleknaz Nayebi, Mahshid Marbouti, Rachel Quapp, Frank Maurer, and Guenther Ruhe. 2017. Crowdsourced exploration of mobile app features: A case study of the fort mcmurray wildfire. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*. IEEE Press, 57–66.
[21] Maleknaz Nayebi and Guenther Ruhe. 2017. Optimized functionality for super mobile apps. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 388–393.
[22] Charith Perera, Chi Harold Liu Member, Srimal Jayawardena, and Min Chen. 2015. Context-aware computing in the internet of things: A survey on internet of things from industrial market perspective. *arXiv preprint arXiv:1502.00164* (2015).
[23] Ronald W Perry. 2007. What is a disaster? In *Handbook of disaster research*. Springer, 1–15.
[24] Raimundo Real and Juan M Vargas. 1996. The probabilistic basis of Jaccard's index of similarity. *Systematic biology* 45, 3 (1996), 380–385.
[25] Martin P. Robillard, Walid Maalej, Robert J. Walker, and Thomas Zimmermann. 2014. *Recommendation Systems in Software Engineering* (2014 ed.). Springer Berlin Heidelberg.
[26] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.
[27] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. 2012. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies* 5, 4 (2012), 318–335.
[28] Vaninha Vieira, Patricia Tedesco, and Ana Carolina Salgado. 2011. Designing context-sensitive systems: An integrated approach. *Expert Systems with Applications* 38, 2 (2011), 1119–1138.
[29] Norha M Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. 2018. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems* 140 (2018), 173–200.
[30] Jared Wade. 2012. Using mobile apps in disasters. *Risk Management* 59, 9 (2012), 6–8.
[31] Jun Ye. 2011. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Mathematical and computer modelling* 53, 1-2 (2011), 91–97.