Mobile App Privacy and Security Recommendation Systems

Dillon V. Stenberg
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
stenb061@morris.umn.edu

ABSTRACT

With the rapid increase of smart phones, the number of mobile applications (Apps) have increased dramatically. Existing recommendation systems have recommended Apps to users based on the popularity of an App or based on the user's interest in other Apps. However, these recommendation systems do not take into account the user's privacy and security preferences. In this paper, I looked at two different recommendation systems that considers user privacy and security preferences.

Keywords

mobile apps, privacy, security, recommendation systems

1. INTRODUCTION

Over the years, there has been a rapid growth of smart phone use and because of this growth a huge number of mobile Apps have been developed for mobile users. At the end of July 2015, there were over 1.5 million Apps on Google Play and as of July 2013 there were over 50 billion downloads. However, a lot of mobile Apps are poorly understood when it comes to activities and functionality related to privacy and security. Mobile Apps, like location service and social service Apps, usually involve permissions that require access to the user's personal data, such as location and the contact lists.

There are some Apps that may require certain access to your device that has nothing to do with the App. Some Apps (i.e. Games) will require location, so that they can transfer that data to third party ad libraries so they can give you targeted advertisements based on your location [3]. People can become very sensitive to the fact that an App can access certain private information that they feel is invasive. Specifically on Android devices, Apps require certain access to security permissions such as reading your contacts or knowing your location. One survey revealed that 54% of U.S. mobile App users, that were surveyed, would not download an App based on the required security permissions for that App, and 30% of App users uninstalled an App after learning what kinds of personal information some Apps were collecting [1]. There are many systems out there that recommend mobile Apps to mobile users based upon popularity or functionality, but none of them recommended Apps based off a user's privacy and security preferences.

This paper describes two recommendation systems that use the user's privacy and security preferences when recommending Apps. The first recommendation method (SPAR) considers popularity and privacy and security when recommending Apps, while the second recommendation considers both the functionality of an App as well as the privacy and security preferences of the user. The focus of the methods are on Android Apps, because the Android system is a permission-based framework.

UMM CSci Senior Seminar Conference, December 2015 Morris, MN.

2. BACKGROUND

Recommendation systems are used in most of today's digital marketplaces. Websites like Amazon and Netflix use recommendation systems to help recommend different products to consumers based on what they rate, what they watch, and what they buy. The Google Play and Apple App store both recommend Apps to you based on what you have bought and downloaded. However, these recommendation systems do not recommend apps based on users' preference of an App's privacy and security.

2.1 Privacy and Data Permissions

On Android devices, Apps require access to certain privacy and data permissions (i.e accessing location and internet) on your mobile device. A permission is related to a critical resource (e.g. Internet, contact, location) on the mobile device. By downloading that App, you are granting it permission to either read or write the corresponding resource. Some of these permissions could be considered as invasive to one's privacy. People may not download an App after knowing what permissions are required to be accessed such as the Facebook mobile App, which requires you to allow it to read and store your contacts from your phone onto your Facebook account [4].

2.2 Latent Matrix Factorization

In previous recommendation systems [2, 6], matrix factorization methods were used to recommend certain products or entertainment to consumers/users. A *latent factorization model* is a technique to recommend products to users. Websites like Amazon and Netflix use this method when recommending products or shows to consumers. Matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. A latent variable is a variable that is not directly observed (directly measured), but is inferred from other variables that are observed. For example, if a user rates a product highly, the system will

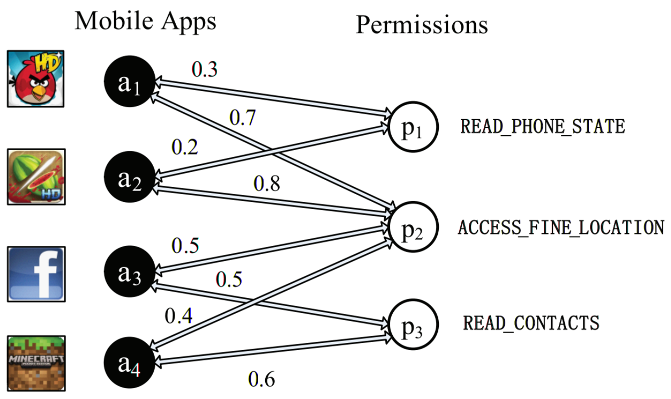


Figure 1: An example of the App permission bipartite graph [11]

look at other users who have also rated that same product highly. From that rating, the system will look at other products that the other users rated highly and recommend those products to the original user [6]. In the study [7], a latent factorization model is created and used when recommending Apps based on their functionality, as well as users' privacy preferences.

2.3 Probability Distributions

Probability distributions are useful by finding the "shape" of the data given or of future possible data and they describes possible values and their probability. In the study [7], they look over the data given from the users' ratings on Apps to help model one user profile latent factor $\mathbf{u}_i \in \mathbb{R}^K$, which is a K-dimension vector in which probability distributions are used. The Gaussian distribution is typically "bell-shaped" curves and is more generalized to higher dimensions. The Poisson distribution has an increasing or decreasing shaped curve and is a discrete frequency distribution that calculates the probability of independent events occurring in a fixed time over a distance (K-dimension).

3. SPAR

A type of recommendation method known as **S**ecurity and **P**rivacy aware mobile **A**pp **R**ecommendation (SPAR), recommends Apps to users while considering both the popularity of an App and the users' privacy preferences. The method uses a flexible approach that is based on *modern portfolio theory* for recommending Apps by balancing the Apps' popularity and the users' privacy preferences. SPAR first estimates a risk score for each App based on security permissions. Second, a risk score is assigned to an App, then each App is ranked by their risk score and popularity. Finally, an *App hash tree* is created to efficiently recommend Apps.

3.1 Estimating Risk Score

SPAR generates a risk score to reflect the security level of an App, where the lower the score, the safer the App is to use. The score is generated by looking directly at the data permissions of each App that is requested. However, there are challenges that must be faced when evaluating the risk score. First, it is difficult to define a risk function with respect to different permissions for evaluating risk scores for Apps, since a lot of permissions are ambiguous and are poorly understood [11]. For example, some permissions that could be considered dangerous (i.e. location) are commonly used in the Apps of some categories (i.e. navigation Apps). However, there are some App categories (i.e. games) that use location to send to ad libraries, so they can better target you with advertisements. Second, the latent relationships between Apps and permissions should be considered, because some Apps will have the same risk score. Finally, a scalable approach should be developed to refine the risk scores, since external knowledge can be leveraged for evaluating potential risks of Apps. To deal with these challenges, Zhu et al. suggests an approach based on a bipartite graph, which can connect each App to the permission automatically without using any predefined risk function. The *App-permission bipartite graph* is used to build the connections between Apps and permissions.

Figure 1 shows an example of an App-permission bipartite graph. In the figure, it shows a set of Apps (Angry Birds, Facebook, etc.) being connected to different permissions, but some of the Apps shown have the same permissions as other Apps. The weight w_{ij} can be estimated by the permission records of all Apps in a_i's category c (i.e. games, tools) where f_{ij} is the number of Apps in category ($a_i \in c$) and E is the edge set, requesting permission p_j:

$$w_{ij} = \frac{f_{ij}}{\sum_{e_{ik} \in E} f_{ik}}$$

To estimate App risk scores with the App-permission bipartite graph, two risk scores *Risk*(a_i) (objective App risk score) and *Risk*(p_j) (global permission risk score) for nodes $a_i \in V^a$ and $p_j \in V^p$, where V^a denotes the set of Apps and V^p denotes the set of permissions, are computed.

3.2 Assigning Risk Score to Apps

In SPAR, a probabilistic approach called Naive Bayes with information Priors (PNB) is used for assigning the risk scores to each App. PNB is a technique that constructs classifiers: models that assign class labels to problem instances [10]. In this case, we are assigning risk scores to Apps. This model is based on Bayes theorem, which describes the probability of an event based on conditions that might be related to the event. In this model, $P(p_j|\theta)$ has a parameter θ that is assumed to follow the Beta prior $Beta(\theta; \alpha_0, \beta_0)$. The Beta prior distribution is a family of continuous probability distributions defined on the interval $[\alpha, \beta]$ [8]. The Beta distribution calculates the probability based on previous "evidence" related to the event. The probability can be estimated by

$$P(p_j|\theta) = \frac{\sum_i^M x_{i,j} + \alpha_0}{M + \alpha_0 + \beta_0},$$

where M is the total number of Apps and $x_{x,j}$ is a binary function which is equal to 1 (requests the permission p_j) or 0 (a_i does not request the permission p_j). PNB also defines three categories of permissions: normal permission, dangerous permissions, and signature/system permissions (e.g. permission to delete Apps) which each category has a specific $Beta(\theta; \alpha_0, \beta_0)$.

3.3 Ranking System

Once the risk score for each App is computed, the Apps are ranked in ascending order based on their risk scores. Moreover, if some of the Apps have the same risk scores, they will be ranked based on their popularity scores (overall rating from users). The Apps are put into clusters where they have the same security level (e.g. low or high). However, by doing this, it is not possible to get an accurate and appropriate segmentation of Apps with respect to their risk

Algorithm 1 Automatic Detection of Security Levels

```
Input: The set of Apps $A = \{a_i\}$; Parameter $\delta$;
Output: The set of security levels $\Psi$;
 1: Rank $A$ in descending order according to Risk(a);
 2: L=$\emptyset$;
 3: for each $i \in [1, |A|]$ do
 4:     $A^* = L \cup \{A[i]\}$;
 5:     calculate $CV(A^*) > \delta$ in terms of Risk(a) $(a \in A^*)$;
 6:     if $(CV(A^*) > \delta)$ then
 7:         $\Psi \cup = L$; $L = \emptyset$ is a new level;
 8:     else
 9:         $L \cup = \{A[i]\}$;
10:     end if
11: end for
12: return $\Psi$
```

scores [11]. To solve this problem, an algorithm, Coefficient of Variation (CV), is developed to automatically segment Apps. Two adjacent Apps in the ranked list are assigned different security levels if their risk scores have dramatic differences. CV determines whether the difference of risk score will put the App in one security level or another. Algorithm 1 shows the specifics, where the parameter δ is a threshold to specify the CV difference required to assign adjacent Apps to different security levels. The risk scores of Apps a_i are put into the algorithm where it loops through the set of Apps A, and checks whether the risk score of a is greater than the parameter δ. If the risk score is greater than δ, the App is put into a new security level. If the risk scores are close enough to the previous scores, then the Apps are put into the appropriate security level. The lower the security level, the higher the security risk.

Apps are now able to be recommended to users: given a specific security level L^* and a category c, Apps will be treated in category c with security $L \geq L^*$ as candidates. There are three types of ranking principles for recommending Apps:

- ***Security Principle:*** The App candidates are first ranked in ascending order by their risk scores and if they have the same scores, they will be ranked further by popularity scores
- ***Popularity Principle:*** The App candidates are ranked in descending order by their popularity scores (e.g. overall rating), and Apps with the same popularity scores are then ranked by their risk scores.
- ***Hybrid Principle:*** It helps balance a user's security preferences and an App's popularity, which is based on the modern portfolio theory.

The modern portfolio theory is originally theorized in the problem of investment of the financial market. An example would be that an investor wants to select a portfolio of n stocks with a fixed investment budget, which, in turn, will return the maximum future return with the minimal risk [5]. So instead of stocks, they are Apps and the future return and risk would be the popularity of the App and security risk.

After ranking Apps with respect to the three different principles, the Apps are organized with respect to their security levels and categories. They are then put into a data structure for App retrieval, namely an *App hash tree*. A *hash tree* is a tree that has labelled nodes of values that have child nodes in different hierarchies. This tree contains two hierarchies: a category level and a security level. For each node in the tree, it contains a hash table that stores the index of an App. The reason why a hash tree is used is so Apps can be divided into different categories (i.e. tools, games) and into different security levels.

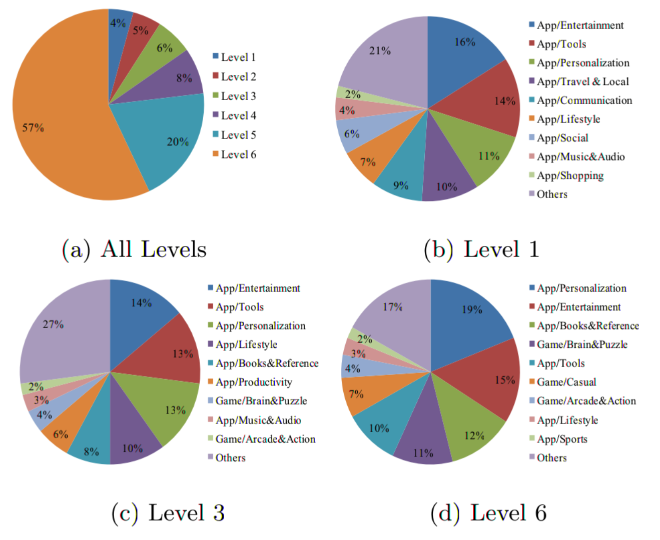
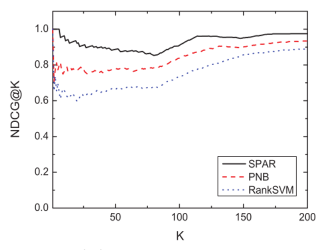


Figure 2: Percentage of (a) Apps at each security level and (b)-(d) App categories at different security levels 1, 3, and 6 [11]

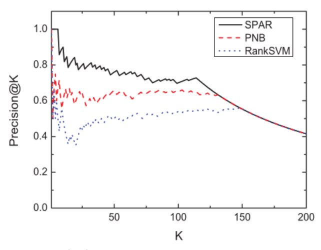
3.4 Experimental Results

The experimental data that was collected in 2012 from Google Play included 170,753 Apps in 30 App categories, and the Apps have 173 unique security access permissions. The Apps were categorized into six segmented security levels (1-6). To organize each App into their respective risk level, the risk scores in Algorithm 1 set $\delta = 0.01 * CV(A)$, where $CV(A)$ is the Coefficient of Variation (CV) of all App risk scores. Figure 2 (a) shows the percentage of Apps inside of each risk level (6 is the most secure and 1 is the most insecure) where most of the Apps fall under level 6 and levels 1-4 are relatively even. This shows that most Apps are secure while only a few Apps have security risks. (b)-(d) shows the percentage of App categories at levels 1, 3, and 6. The figure shows what Apps have more permissions, such as the categories "Tools", "Travel and Local", and "Communication", and therefore are more likely to have potential risks.

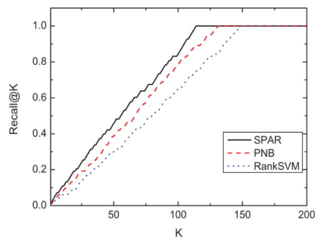
The recommendation method SPAR is compared to two different methods of ranking App risks: Naive Bayes with information Priors (PNB) and RankSVM. RankSVM is learning-to-rank approach that analyzes data and recognizes patterns by the relationship of a specific query. 200 secure and 200 insecure Apps were used as training data. From all those Apps, Zhu et al. selects the top 100 ranked Apps (most insecure) and the bottom 100 ranked Apps (most secure) and merges them into a pool of unique Apps, that totalled to 496 Apps, into a data set. Three users were brought to manually label each App with a score of 2 (insecure), 1 (not sure), and a 0 (secure). Each user gave their own judgement on the App based on their experience with it (i.e. downloading and using the App). Then each App was given a judgement score $f(a) \in [0, 6]$. Finally, the 496 Apps were then ranked by each method and were made into three ranked lists of Apps. Then a popular metric evaluation tool was used: Normalized Discounted Cumulative Gain (NDCG) to determine the ranks of performance between each result. The DCG given



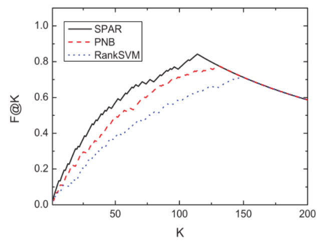
(a) $NDCG@K$



(b) $Precision@K$



(c) $Recall@K$



(d) $F@K$

Figure 3: Overall performance of the different methods based on user judgement [11]

a cut-off rank K is calculated by

$$DCG@K = \sum_{i=1}^{K} \frac{2^{Rel(a_i)-1}}{\log_2(1+i)},$$

where $f(a_i)$ is the App a's assigned judgment score ($f(a) \in [0,6]$) and $Rel(a_i) = f(a_i)$ is the judgement score. K is the number Apps being recommended. Figure 3 shows the overall performance between the three methods by four different information retrieval metrics [9]:

- **NDCG** measures the performance of a recommendation system based on the graded relevance of the recommended entities.
- ***Precision@K*** is the proportion of retrieved instances that are relevant to the query and ignores anything below the cut-off rank K.
- ***Recall@K*** is the proportion of relevant instances that are retrieved successfully.
- ***F@K*** is the balance of precision and recall.

Figure 3 shows the results of the three methods compared with respect to the four different evaluation metrics. It can be seen that the method SPAR outperforms the other two methods: PNB and RankSVM, over all four evaluation metrics and the improvement is more significant when K is smaller. Since SPAR uses PNB for assigning risk scores to Apps, if PNB had used the bipartite graph as described in this section, its overall performance would improve according to Zhu et al. Both SPAR and PNB outperform RankSVM in every evaluation metric. Learning-to-rank approaches are not as effective for estimating App risks, because they mainly rely on the effectiveness of feature extraction [11]. Overall, SPAR performed better when estimating risk scores for Apps.

4. PRIVACY-RESPECT APP RECOMMENDATION MODEL

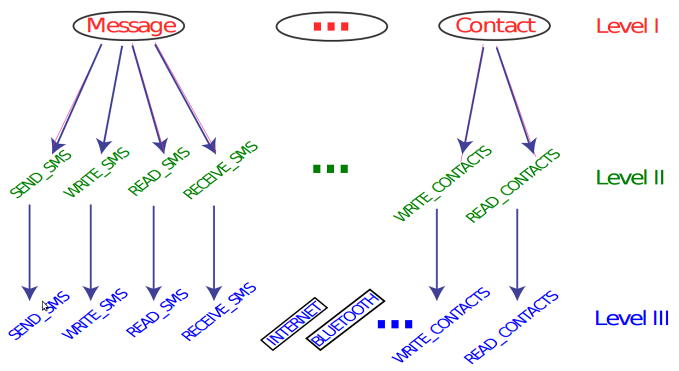


Figure 4: Illustration of the three privacy levels [7]

This section presents a proposed user privacy-respect App recommendation model with App functionality in mind. A new latent factorization model is constructed to capture the trade-off between functionality and user privacy preference. Three levels of privacy information are used to characterize users' privacy preferences. For testing their model, Liu et al. crawls a real-world dataset (collection of data) from Google Play and uses it to evaluate the model as well as other previous methods [7].

4.1 User Privacy Levels

Each user has their own preference of privacy and security. This method defines three privacy levels when determining a user's privacy concern, which each consists of a set of resources and their corresponding security permissions. Liu et al. compares each of the different privacy levels when testing. The privacy levels are also used in determining the privacy respect score in 4.2. Figure 4 describes the mapping of each level.

Level I: This level consists of 10 resources (including location, contact, message) as private. However, this level does not access the security permissions that are associated with each resource (security permission).

Level II: This level consists of the same 10 resources as in Level I, but can distinguish the different security permissions that are associated with that resource. In this level, a user can be flexible when choosing what sort of privacy concern is more important to them. In total, this level is expressed by the 10 resources and a total of 23 security permissions.

Level III: This level consists of all resources and distinguishes the associated security permissions from level I and II as well as other resources (e.i. Internet and bluetooth) on a mobile device. This level is more complete and can express all dangerous Android permissions. In total there are 72 security permissions.

4.2 Constructing a Latent Factorization Model

The aim of this method is to quantify the trade-off of both App functionality and user privacy preference [7]. The study introduces $g_{\mathrm{func,i,j}}$ as the *functionality match score* of the interest of user i and functionality of App j and $g_{\mathrm{privacy,i,j}}$ is the *privacy respect score* of the privacy preference of user

i and the privacy information used by App j.

Functionality match model: By using latent factor models used in standard recommendation systems [6], a model is created based on a user j's interest as a latent vector $\mathbf{u}_i^{\text{interest}} \in \mathbb{R}^K$ and an App j's functionality as a latent vector $\mathbf{v}_j \in \mathbb{R}^K$, where K is the number of latent dimensions of user interest and App functionalities. Then the *functionality match score*, $g_{\text{func,i,j}}$, is modeled as:

$$g_{\text{func,i,j}} = f\left(\mathbf{u}_i^{\text{interest}}, \mathbf{v}_j\right),$$

Privacy respect model: A latent factor model is also used to describe user privacy preference and App's private information. A user i's privacy preference is denoted as a latent factor $\mathbf{u}_i^{\text{privacy}} \in \mathbb{R}^K$. Each privacy information is modeled in the set of S at a given privacy level as a privacy latent factor $\mathbf{p}_s \in \mathbb{R}^K$. Then the *privacy respect score*, $g_{\text{privacy,i,j}}$, is modeled as:

$$g_{\text{privacy}} = f\left(\mathbf{u}_i^{\text{privacy}}, \sum_{S \in \prod_j} \mathbf{p}_s\right),$$

where $\prod_j$ is the set of privacy information of the App j.

Trade-off between functionality and privacy: A model that contains a user i's overall functionality preference for an App j, which is denoted by matrix $g_{i,j}$, is the sum of the functionality match score and the privacy respect score, we have:

$$g_{i,j} = g_{\text{func,i,j}} + \lambda g_{\text{privacy,i,j}}$$

where λ is a coefficient used to balance App functionality and user privacy preference.

4.3 Model Specifications

This section presents a detailed model specification for the latent factorization model. In the previous subsection, the representation of user interests and user privacy preferences being two latent factors, both the privacy preference and user interest vectors are combined into one user profile latent factor $\mathbf{u}_i \in \mathbb{R}^K$. Combining them can reduce parameters to learn and will improve computational efficiency [7]. In each App j, the user profile latent factor is modeled by a functionality latent factor and a privacy latent factor as $\mathbf{v}_j + \lambda \sum_{s \in \prod_j} \mathbf{p}_s$, where $\prod_j$ is the privacy information set for App j and $\mathbf{p}_s$ is the privacy latent factor. So a user i's preference score (rating), x_{ij}, for an App j is represented by

$$x_{ij} = \mathbf{u}_i^T \left(\mathbf{v}_j + \lambda \frac{1}{|\prod_j|} \sum_{s \in \prod_j} \mathbf{p}_s\right)$$

where $\frac{1}{|\prod_j|}$ is there for each App to adjust the unbalanced number of privacy information.

To model user profile and App profile, the user-App preference score (rating) x_{ij} for an App is followed by a probability distribution $\Pr(y_{ij}|x_{ij}, \theta)$, that can infer the latent factors $\mathbf{u}_i$, $\mathbf{v}_j$, and $\mathbf{p}_s$. The data gathered does not follow Gaussian distribution, so Poisson distribution was used instead to approximate the data distribution, which is denoted by $y_{ij} \sim \text{Poisson}(x_{ij})$ [2]:

$$Pr(y_{ij}|x_{ij}) = (x_{ij})^{y_{ij}} \frac{exp(-x_{ij})}{y_{ij}!}.$$

Poisson distribution is a better choice for modeling discrete user-item responses [7]. It is better at capturing real user-item response data by setting non-negative constraints on latent factors. It forces response variables to be in a wider range than the rating based response, thus capturing a better preference order. Also, only the observed part of the user-item matrix needs to be iterated during modeling, which provides an advantage for the sparsity or user-item matrix in recommendation problems [7].

4.4 Experimental Results

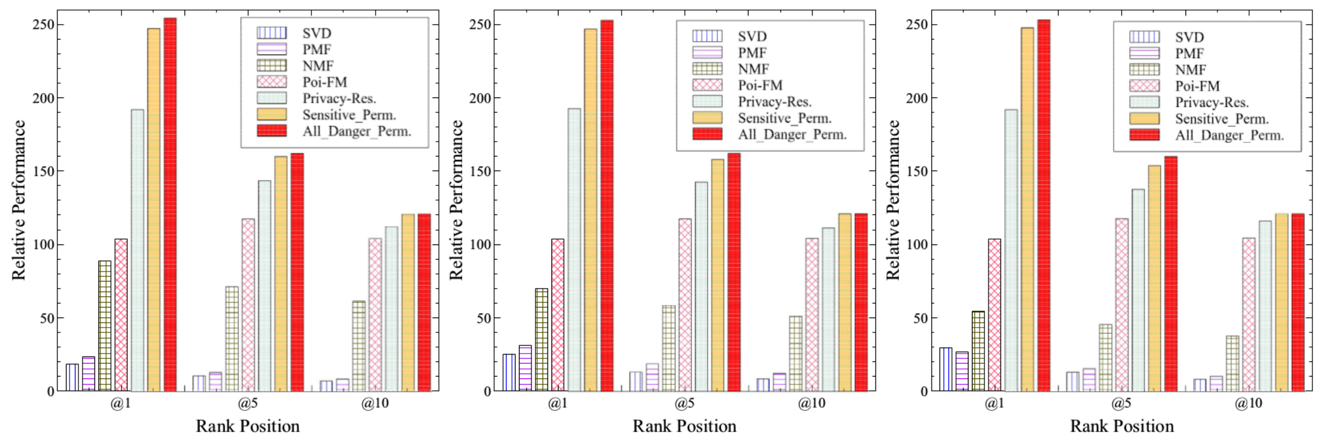
Data for the experiments were collected from Google Play. Since a user's ratings are publicly available, the Google ID of a user can be seen and can locate all Apps the user has rated. A list of users was obtained and a crawler was written to retrieve all rated Apps of those users in that list. For each App that was retrieved, they crawled its permissions from Google Play. The crawls were performed between June and July 2014. The dataset included 16,344 users, 6,157 Apps, and 263,054 rating observations. Of the rated Apps, 80% of them were used only for training data, while the rest of the Apps were used for actual testing. Seven different approaches were used when conducting experiments and comparing the results of them. The proposed model of [7] is compared to four previous latent factor based recommendation models:

- Singular Value Decomposition (SVD) [6]: SVD is a low dimension decomposition based recommendation method.
- Probabilistic Matrix Factorization (PMF): PMF is like SVD, but instead is a probabilistic framework.
- Non-negative Matrix Factorization (NMF): Similar to SVD, but the latent vectors have to be non-negative.
- Poisson Factor Model (Poi-FM) [2]: Poisson factor model is an alternative for latent factor model for different applications.

The latent factorization model that was created by Liu et al. is based on some of these models (i.e. SVD, PMF), however, instead of just looking at user interest in Apps, this method also considers the user's privacy preferences when recommending Apps. Those four previous recommendation models were compared to different variants of the privacy levels that were mentioned in 4.1. The three different privacy variants compared:

- `Privacy_Res`: Privacy-respect App recommendation with the Level I privacy level.
- `Sensitive_Perm`: Privacy-respect App recommendation with the Level II privacy level.
- `All_Danger_Perm`: Privacy-respect App recommendation with the Level III privacy level.

Three of the evaluation metrics mentioned in 3.4: precision@N, recall@N, and F@N, were used to evaluate all seven approaches. Each approach was tested at different latent dimensions such as K = 20, 30, and 50. At each latent dimension, each approach was tested with N Apps such as N = 1, 5, and 10. Figure 5 illustrates the relative performance of the seven approaches over all of the evaluation metrics. The relative performance measures the improvement upon a random recommendation method. By looking at the figure, it shows that the three variants of this sections method:



(a) $K = 20$ **(b)** $K = 30$ **(c)** $K = 50$

Figure 5: Relative performance @N with different latent dimensions K [7]

`Privacy_Res`, `Sensitive_Perm`, and `All_Danger_Perm` outperformed the other four compared approaches. NMF and Poi-FM outperform both SVD and PMF, and that Poi-FM outperforms NMF. There is not much difference in performance when coming comparing each approach to itself on a different latent dimension. When there are more Apps tested for each latent dimension, the relative performance decreases over all approaches.

5. CONCLUSIONS

In this paper, we have analyzed two recommendation methods that consider the privacy and security preferences of users. Both methods considered user privacy and security preferences, while the methods that each method compared to, did not. The Privacy-Respect App modeled itself after the latent factorization model SVD, but considered both functionality and user privacy preferences. SPAR and the Privacy-Respect App are structured differently, but consider user privacy preferences, as well as functionality (Privacy-Respect App) and popularity (SPAR). The Privacy-Respect App's three main variants of the privacy levels (Level I, II, and III) outperformed the other previous latent factorization methods, concluding that Liu et al.'s model created, was more efficient at recommending Apps to users. SPAR had outperformed both PNB and RankSVM in every evaluation metric, concluding that SPAR is more efficient at recommending Apps to users.

6. ACKNOWLEDGMENTS

Thank you to Nic McPhee, Elena Machkasova, Kristin Lamberty, Peter Dolan, Chase Ottomoeller, Ashton Stenberg, and Sarah Drenner for their invaluable feedback.

7. REFERENCES

[1] S. A. Boyles, Jan Lauren and M. Madden. Privacy and data management on mobile devices, 2012. `http://www.pewinternet.org/2012/09/05/privacy-and-data-management-on-mobile-devices/,[Online;accessed18-September-2015]`.

[2] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with poisson factorization. *CoRR*, abs/1311.1704, 2013.

[3] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[4] S. Grobart. The facebook scare that wasn't, 2011. `http://gadgetwise.blogs.nytimes.com/2011/08/10/the-facebook-scare-that-wasnt/,[Online;accessed18-September-2015]`.

[5] Investopedia. Modern portfolio theory, 2015. `http://www.investopedia.com/walkthrough/fund-guide/introduction/1/modern-portfolio-theory-mpt.aspx,[Online;accessed1-November-2015]`.

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[7] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, and H. Xiong. Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 315–324, New York, NY, USA, 2015. ACM.

[8] Wikipedia. Beta distribution — wikipedia, the free encyclopedia, 2015. `https://en.wikipedia.org/w/index.php?title=Beta_distribution&oldid=688314127,[Online;accessed6-November-2015]`.

[9] Wikipedia. Information retrieval — wikipedia, the free encyclopedia, 2015. `https://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=685819601,[Online;accessed6-November-2015]`.

[10] Wikipedia. Naive bayes classifier — wikipedia, the free encyclopedia, 2015. `https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=685075040,[Online;accessed6-November-2015]`.

[11] H. Zhu, H. Xiong, Y. Ge, and E. Chen. Mobile app recommendations with security and privacy awareness. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 951–960, New York, NY, USA, 2014. ACM.