Scrutinizing Mobile App Recommendation: Identifying Important App-related Indicators

Jovian Lin[1], Kazunari Sugiyama[1], Min-Yen Kan[1,2], and Tat-Seng Chua[1,2]

[1] School of Computing, National University of Singapore,
[2] Interactive and Digital Media Institute, National University of Singapore, Singapore
jovian.lin@gmail.com, sugiyama@comp.nus.edu.sg
kanmy@comp.nus.edu.sg, chuats@comp.nus.edu.sg

Abstract. Among several traditional and novel mobile app recommender techniques that utilize a diverse set of app-related features (such as an app's Twitter followers, various version instances, *etc*.), which app-related features are the most important indicators for app recommendation? In this paper, we develop a hybrid app recommender framework that integrates a variety of app-related features and recommendation techniques, and then identify the most important indicators for the app recommendation task. Our results reveal an interesting correlation with data from third-party app analytics companies; and suggest that, in the context of mobile app recommendation, more focus could be placed in user and trend analysis via social networks.

Keywords: Recommender systems, Mobile apps, Gradient tree boosting

1 Introduction

Traditional recommendation approaches either learn a user's preference from their ratings (*i.e.*, collaborative filtering) or the contents of previously-consumed items (*i.e.*, content-based filtering). Despite the pervasive use of collaborative filtering in several domains such as books, movies, and music, its effectiveness is hindered by insufficient ratings, particularly towards newly-released items — a problem that is commonly known as the "cold-start." Moreoever, due to noisy and unreliable descriptions of apps, content-based filtering does not work well in the app domain [10].

With the widespread interest and pervasiveness of mobile apps, several novel recommendation techniques that take advantage of the unique characteristics of the app domain have emerged. The first type focuses on collecting additional internal information from the user's mobile device, which analyzes the usage behavior of individual apps via anonymized network data from cellular carriers [18] as well as usage patterns of users via their in-house recommender systems [19, 6, 1]. The second type makes use of external information such as spatial data from GPS sensors to provide context-aware app recommendations [22, 7]. These two types, however, rely on data that is generally difficult to obtain, causing the secondary problem of data-sparsity. On the contrary, the third type consists of works that capitalize on more unique characteristics of the app domain that may not be applicable to other domains. For instance, "follower" information of an app's Twitter account was used to substitute missing user ratings [10], which proved to be useful in cold-start situations. Another work tried to find the likelihood of which a current app would be replaced by another [20]. Alternatively, by taking the fact that apps change and evolve with every new version update, a "version-sensitive" recommendation technique was constructed to identify desired functionalities (from various version descriptions of apps) that users are looking for [11].

With a variety of app recommendation techniques utilizing different sources of information, of which some may be available while others are not (*e.g.*, not all apps have user ratings), we explore the advantages of a hybrid app recommendation framework that combines traditional and novel techniques. More importantly, through the hybrid framework, we seek to identify the most important app-related indicators for the recommendation task.

The steps are as follows: First, using gradient tree boosting (GTB) [8], several recommendation techniques and their information sources are integrated to form a hybrid app recommender framework. After that, we further look into each component of the feature set to find the most significant features in the hybrid framework. Our findings show an interesting correlation with data from third-party app analytics companies, and suggest that, in the context of mobile app recommendation, more focus could be placed in user and trend analysis via social networks.

2 Related Work

2.1 Mobile App Retrieval

Chen *et al.* [5] proposed a framework for detecting similar apps by constructing kernel functions based on multi-modal heterogeneous data of each app (description text, images, user reviews, and so on) and learning optimal weights for the kernels. They also applied this approach to mobile app tagging [4]. While Chen *et al.*'s work utilized different modalities of an app, Park *et al.* [14] exclusively leveraged text information such as reviews and descriptions (written by users and developers, respectively) and designed a topic model that can bridge vocabulary gap between them to improve app retrieval. Zhang *et al.* [21] developed a mobile query auto-completion model that exploits installed app and recently opened app. In addition, Martin *et al.* [13] has published a nice survey on app store analysis that identifies some directions for software engineering such as requirements engineering, release planning, software design, testing, and so on.

2.2 Mobile App Recommendation

In order to deal with the recent rise in the number of apps, works on mobile app recommendation are emerging. Some of these works focus on collecting additional information from the mobile device to improve recommendation accuracy. Xu *et al.* [18] investigated the diverse usage behaviors of individual apps by using anonymized network data from a tier-1 cellular carrier in the United States. While Yan and Chen [19], Costa-Montenegro *et al.* [6], and Baeza-Yates *et al.* [1] analyzed internal information such as the usage patterns of each user to construct app recommendation system, Zheng *et al.* [22] and Davidsson and Moritz [7] utilized external information such as GPS sensor information to provide context-aware app recommendation. Lin *et al.* [10] utilized app-related information on Twitter to improve app recommendation in cold-start situations. Their subsequent work focused on app's uniqueness of version update, and then proposed an app recommendation system that leverages version features such as textual description of the changes in a version, version metadata [11]. These two works are compiled into [9]. Yin *et al.* [20] considered behavioral factors that invoke a user to replace an old app with a new one, and introduced the notion of "actual value" (satisfactory value of the app after the user used it) and "tempting value" (the estimated satisfactory value that the app may have), thereby regarding app recommendation as a result of the contest between these two values. Zhu *et al.* [23] and Liu *et al.* [12] incorporated both each user's interest and privacy preferences to provide app recommendation as apps could have privileges to access the user's sensitive personal information such as locations, contacts, and messages. While the aforementioned works recommend apps that are relevant to each user's interests, Bhandari *et al.* [2] proposed a graph-based method for recommending serendipitous apps.

3 Methodology

3.1 Feature Set

Inspired by Wang *et al.*'s work [17], the features that we use can be categorized into the following three distinct groups:

1. the app's <u>m</u>arketing-related <u>m</u>etadata ($\mathbb{M}$),
2. the user's <u>h</u>istory-related information ($\mathbb{H}$), and
3. the <u>r</u>ecommendation scores of different recommender systems ($\mathbb{R}$).

As illustrated in Figure 1, every candidate app's feature vector $\boldsymbol{X_{u,a}}$ is composed of all three groups of information: $\boldsymbol{X_{u,a}} = \{\boldsymbol{X_a^M}, \boldsymbol{X_{u,a}^H}, \boldsymbol{X_{u,a}^R}\}$ where $\boldsymbol{X_{u,a}}$ represents the feature vector of the app a for user u, while $\boldsymbol{M}$, $\boldsymbol{H}$, and $\boldsymbol{R}$ represent the features from the users' history, apps' metadata, and recommendation scores from various recommendation techniques, respectively.

3.1.1 App's Marketing-related Metadata ($\mathbb{M}$)

The features here pertain to the app's metadata or marketing-related information. We include most of the components of an app's official metadata from the iTunes App Store, such as the various genres that the app is assigned to, its price, average ratings, *etc*. We also include external information, particularly ubiquitous data from social networks, such as the number of versions an app has, the number of Facebook "likes" it has (zero if the app has no Facebook handle), and the number of Twitter followers it has (zero if the app has no Twitter handle). The *blue* components in Figure 1 show all the information of an app's marketing-related features.

3.1.2 User's History-related Information ($\mathbb{H}$)

User history is primarily extracted from the rating history of users, and it is a crucial component for the purpose of providing *personalized* recommendations. In addition, inspired by Wang *et al.*'s method [17] for generating additional user metadata by scrutinizing the genres of items that users have consumed, we also consider the user's preference of each app genre g. For instance, a user might be a loyal consumer of the "games" genre, yet not in the "food & drink" genre. We thus include the number of times (*i.e.*, the "count") that apps in genre g were consumed by user u (represented in *green* in Figure 1).

3.1.3 Recommendation Scores from Different Recommender Techniques ($\mathbb{R}$)

We also include the recommendation scores generated from four recommendation techniques: *i)* collaborative filtering, *ii)* content-based filtering, *iii)* "Twitter-follower-based app recommendation" (TWF) [10], and *iv)* "version-sensitive recommendation" (VSR) [11]. These are represented by the red components in Figure 1.

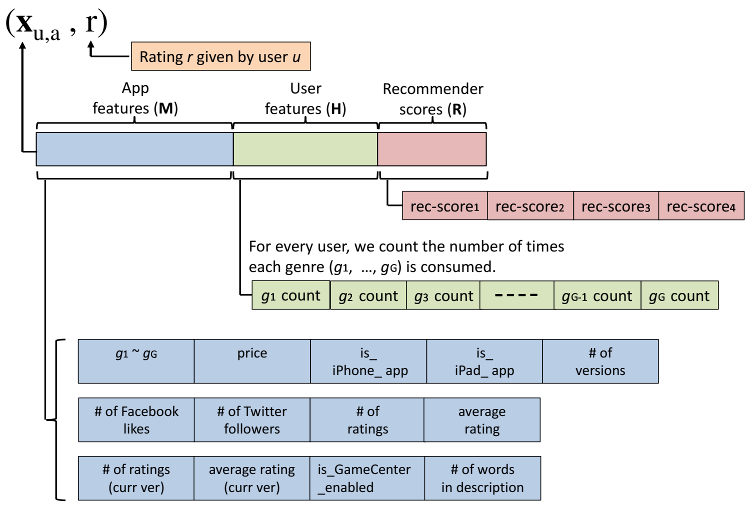

Fig. 1. An app's feature vector $(\boldsymbol{X_{u,a}}, r)$, which contains app features, user features, the various recommendation scores, and the user's rating.

We employ probabilistic matrix factorization (PMF) [15] to implement collaborative filtering as it is a state-of-the art technique that models the user-item ratings matrix as a product of two lower-rank user and item matrices, and it has been used in many previous recommendation works due to its highly flexibility and extendability. We also employ latent Dirichlet allocation (LDA) [3] to implement content-based filtering (on apps' textual descriptions) as it effectively provides an interpretable and low-dimensional representation of the items. In addition, we select TWF and VSR due to their ability to make use of ubiquitous information from Twitter's API and version data from third-party app analytics companies, respectively. With the hybrid app recommendation that is modeled by gradient tree boosting (GTB) [8], we further look into each component of the feature set (*i.e.*, $\boldsymbol{M}$, $\boldsymbol{H}$, and $\boldsymbol{R}$) in the hybrid model based on relative influence[3].

3.2 Combining App Features

Inspired by BellKor's winning solution for the Netflix Prize[4], we turn to Gradient Tree Boosting (GTB), a machine learning algorithm that iteratively constructs an ensemble of weak decision tree learners through boosting [8]. It produces an accurate and effective off-the-shelf procedure for data mining that can be directly applied to the data without requiring a great deal of time-consuming data preprocessing or careful tuning of the learning procedure.

To generate recommendations, the learned GTB predicts the rating that a user may give to an app. After which, it ranks all recommended apps in descending order of rating

[3] Friedman [8] proposed the relative influence for boosted estimates to reflect each feature's contribution of reducing the loss by splitting on the feature.

[4] Y. Koren: "The BellKor Solution to the Netflix Grand Prize," http://www.stat.osu.edu/ dmsl/GrandPrize2009_BPC_BellKor.pdf

to produce a ranked list for each user. Here, we use a popular Python machine learning package from scikit-learn[5] to implement GTB.

4 Experimental Setup

We construct our experimental dataset by crawling the information on Apple's iTunes App Store[6] (app metadata, users, and ratings), App Annie[7] (version information of apps), Twitter (for the Twitter followers of apps), and Facebook (for the "likes" information of apps). Our dataset includes 33,802 apps, 16,450 users, and 3,106,759 ratings after we retain only unique users who give at least 30 ratings. Among the 33,802 apps, 7,124 (21.1%) have Twitter accounts, 9,288 (27.5%) have Facebook accounts, and 10,520 (31.1%) have at least five versions. Note that 678 (2.0%) apps have both Twitter and Facebook accounts. We perform 5-fold cross validation, where in each fold, we take the first 80% of the apps (chronologically) as training data for the individual recommendation techniques, use the following 10% as the training data for the unified model (*i.e.*, the probe set of GTB), and use the remaining 10% for testing.

4.1 Comparative Recommender Systems

We compare two types of recommender systems: individual and hybrid. For individual systems which are baselines, we implement the four recommender algorithms mentioned in Section 3.1.3, namely, collaborative filtering (PMF) [15], content-based filtering (LDA) [3], TWF [10], and VSR [11]. For the hybrid systems, we create three subsets of the GTB framework using a smaller set of features. That is, on top of our gradient boosting hybrid framework GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$), we create three more hybrid systems: GTB($\mathbb{R}$), GTB($\mathbb{H}$, $\mathbb{R}$), and GTB($\mathbb{M}$, $\mathbb{R}$), where "$\mathbb{M}$", "$\mathbb{H}$", and "$\mathbb{R}$" represent the various information $\boldsymbol{X_a^M}$, $\boldsymbol{X_{u,a}^H}$, and $\boldsymbol{X_{u,a}^R}$ mentioned in Section 3.1, respectively.

Table 1 shows the details of the various recommendation techniques and their feature set. For the individual recommender systems, the feature set contains the user's <u>h</u>istory-related features ($\boldsymbol{X_{u,a}^H}$) that are generated from the user's previous ratings history as well as the app data. The hybrid models further integrate the product's marketing-related <u>m</u>etadata ($\boldsymbol{X_a^M}$) and the <u>r</u>ecommender scores generated by the individual recommender systems ($\boldsymbol{X_{u,a}^R}$).

4.2 Evaluation Metric

Our system ranks the recommended apps based on the probability in which a user is likely to download the app. This methodology leads to two possible evaluation metrics: precision and recall. However, a missing rating in the training set is ambiguous as it may either mean that the user is not interested in the app, or that the user does not know about the app (*i.e.*, truly missing). This makes it difficult to accurately compute precision [16]. But since the known ratings are true positives, recall is a more pertinent measure as it only considers the positively rated apps within the top M, namely, a high recall with a lower M will be a better system. We thus chose Recall@M (especially, $M = 50$) as our primary evaluation metric.

[5] http://scikit-learn.org/stable/modules/ensemble.html (Ver 0.15.0)

[6] https://itunes.apple.com/us/genre/ios/id36?mt=8

[7] https://www.appannie.com/

Table 1. Various recommendation techniques.

Technique	Feature Set
PMF [15]	Collaborative filtering with $\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{H}_{u,a}\}$
LDA [3]	Content-based filtering with $\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{H}_{u,a}\}$
TWF [10]	Twitter-follower recommender with $\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{H}_{u,a}\}$
VSR [11]	Version-sensitive recommendation with $\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{H}_{u,a}\}$
GTB($\mathbb{R}$)	$\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{R}_{u,a}\}$
GTB($\mathbb{H}$, $\mathbb{R}$)	$\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{H}_{u,a}, \boldsymbol{X}^{R}_{u,a}\}$
GTB($\mathbb{M}$, $\mathbb{R}$)	$\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{M}_{a}, \boldsymbol{X}^{R}_{u,a}\}$
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$)	$\boldsymbol{X}_{u,a} = \{\boldsymbol{X}^{M}_{a}, \boldsymbol{X}^{H}_{u,a}, \boldsymbol{X}^{R}_{u,a}\}$

5 Experimental Results

5.1 Individual Recommender Techniques

Figure 2 shows Recall@50 obtained by different recommender systems. Among the individual recommender techniques (*i.e.*, the first four bars from the left), content-based filtering (LDA) achieves the best performance, *i.e.*, it outperforms collaborative filtering (PMF), TWF, and VSR. At first, it is surprising that content-based filtering (LDA) is the best individual technique among the other individual algorithms, especially against state-of-the-art ones. But given that the dataset contains some apps that: *i)* do not have enough ratings for collaborative filtering, *ii)* do not have Twitter accounts (78.9%), and *iii)* do not have sufficient version information (68.9%), it is reasonable that these techniques underperform due to the lack of sufficient information for every app, whereas content-based filtering (LDA) works better because apps always have app descriptions to construct a recommendation model. In other words, in general and practical situations where there are a variety of apps that have and do not have ratings, Twitter accounts, and version information, content-based filtering is the more reliable technique.

5.2 Hybrid Recommender Techniques

Next, we explore the GTB models in Figure 2 (the last four bars). All of our GTB models outperform the individual techniques described in Section 5.1. This is expected as many other works that use GTB, particularly those involved in the Netflix prize, have also reported improvements against individual baselines. We also observe a general improvement in recall when we incorporate more components into the feature set. For example, GTB($\mathbb{M}$,$\mathbb{R}$) and GTB($\mathbb{M}$,$\mathbb{H}$,$\mathbb{R}$) outperform GTB($\mathbb{R}$) and GTB($\mathbb{M}$,$\mathbb{R}$), respectively. We observe an interesting small anomaly, in which GTB($\mathbb{H}$,$\mathbb{R}$) slightly underperforms GTB($\mathbb{R}$), whereas GTB($\mathbb{M}$,$\mathbb{R}$) significantly outperforms both GTB($\mathbb{R}$) and GTB($\mathbb{H}$,$\mathbb{R}$). In other words, the recommendation scores ($\mathbb{R}$) is more effective when it is combined with app metadata ($\mathbb{M}$) than when it is combined with user features ($\mathbb{H}$). This suggests that app metadata ($\mathbb{M}$) complements the feature of recommendation scores ($\mathbb{R}$) — which actually makes sense as, given the assortment of app metadata ($\mathbb{M}$) that coincides with recommendation scores ($\mathbb{R}$), a correlation pattern can be better identified. For example, the app metadata of Twitter followers would complement the recommendation score provided by TWF, while the number of versions would complement the recommendation score generated by VSR; likewise, the number of ratings would complement the recommendation score given by collaborative filtering. On the contrary, as

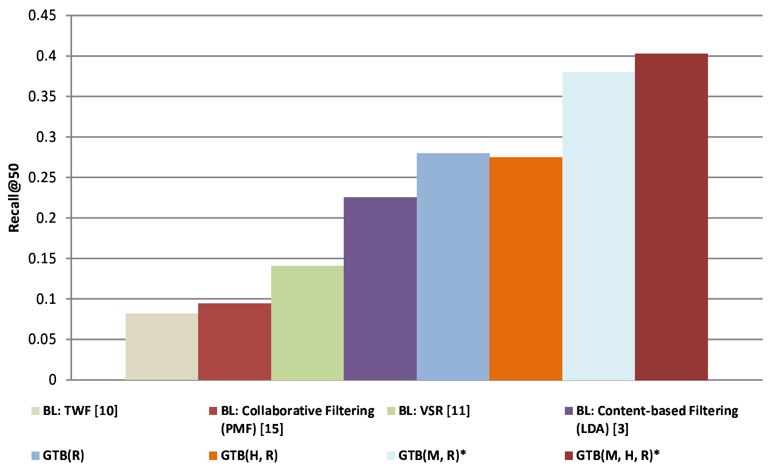

Fig. 2. Recall@50 obtained by individual and hybrid recommender systems. "BL" stands for "baselines." "*" denotes the difference between combined techniques (GTB($\mathbb{M}$,$\mathbb{R}$) and GTB($\mathbb{M}$,$\mathbb{H}$,$\mathbb{R}$)) and the best baseline (content-based filtering (LDA)) is statistically significant for $p < 0.01$.

features from user history ($\mathbb{H}$) mainly consists of the number of times each genre is consumed, it has less obvious correlations.

5.3 Ablation Testing

5.3.1 Ablation Testing for Hybrid Recommendation Techniques

The experimental results described in Section 5.2 show the overall effectiveness of all four combined recommendation techniques as well as user features and app information. To gain a deeper understanding of the individual recommendation techniques, we further perform ablation testing by excluding one of the four recommendation techniques from GTB($\mathbb{M}$,$\mathbb{H}$,$\mathbb{R}$), while at the same time, using the user features and app metadata, $\boldsymbol{X}^{\boldsymbol{H}}_{\boldsymbol{u},\boldsymbol{a}}$ and $\boldsymbol{X}^{\boldsymbol{M}}_{\boldsymbol{a}}$.

Table 2 shows recall@50 obtained by the ablation testing in which we ablate one recommendation technique out of the four. We observe the followings from Table 2:

- Content-based filtering (LDA), which achieves the best recall among all individual baselines, also causes the largest dip in recall when we ablate it from the unifying model. That is, "GTB($\mathbb{H}$,$\mathbb{M}$,$\mathbb{R}$) excluding content-based filtering" has the lowest score (0.237) among the four ablation baselines. This is unsurprising as it is expected when we omit the strongest individual predictor.
- Although VSR individually outperforms collaborative filtering (0.141 against 0.094), ablating it from the unifying model does not have very much impact; in fact, ablating collaborative filtering (PMF) has more impact than ablating VSR.
- It would seem that, from this initial ablation study, both of the traditional recommendation techniques, collaborative filtering (PMF) and content-based filtering (LDA) are more effective than VSR and TWF as the two traditional techniques bring about the two biggest dips in recall when we ablate them.

Table 2. Recall@50 obtained by ablation testing.

Feature	Recall@50
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$)	**0.403**
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$), excluding TWF [10]	0.363
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$), excluding VSR [11]	0.346
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$), excluding Collaborative Filtering (PMF) [15]	0.292
GTB($\mathbb{M}$, $\mathbb{H}$, $\mathbb{R}$), excluding Content-based Filtering (LDA) [3]	0.237
TWF [10]	0.082
VSR [11]	0.141
Collaborative Filtering (PMF) [15]	0.094
Content-based filtering (LDA) [3]	0.225

– However, we should not let this relative ablation comparison undermine the improvements that VSR and TWF have brought about. In fact, VSR and TWF improve recall by 16.5% and 11.0%, respectively. More importantly, by utilizing these unique and less obvious signals in the app domain (compared with other traditional domains in recommender systems), we have gained significant improvements for general app recommendation[8]. In other words, different pieces of evidences (*e.g.*, Twitter followers and versions) that, when present, can be utilized sufficiently to create a discernible improvement in recommendation quality.

Still, this initial ablation testing does not paint a full picture, especially regarding VSR and TWF, as 68.9% of apps do not have sufficient version information while 78.9% of apps do not have Twitter accounts (see Section 4). Therefore, the lack of information does not provide a well grounded conclusion. In order to investigate the real utility of VSR and TWF, we further scrutinize our data by utilizing a subset of data that has sufficient Twitter and version information in the unifying model.

5.3.2 Ablation Testing Using Sufficient Twiter Information

Similar to Section 5.3.1, we also perform ablation testing using a dataset with full Twitter information. Table 3 shows recall@50 obtained by this study where GTB$_{TWF}(\ldots)$ represents the model that uses full Twitter information in our controlled ablation testing. Table 3 indicates the followings:

– Under a dataset with full Twitter information, we observe a reordering of recommendation techniques whereby TWF becomes consequential — ablating it causes the largest dip in recall scores (0.338) for the unifying model.
– Not only does this justify TWF's effectiveness but more importantly, it indicates that when certain evidence is available (here, Twitter followers information), this changes the signals that are used in the unifying model, allowing TWF to displace the traditional, well-established recommendation techniques.

5.3.3 Ablation Testing Using Sufficient Version Information

Furthermore, we perform another ablation testing using a dataset with full version information. Table 4 shows the recall@50 obtained by this study where GTB$_{VSR}(\ldots)$ represents the model that uses full version information in our controlled ablation testing. According to Table 4, we observe the followings:

[8] In fact, on 21 September 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%. That is, US$1M for an improvement of 10.06%.

Table 3. Recall@50 obtained by controlled ablation testing using sufficient **Twitter** information.

Feature	**Recall@50**
$\text{GTB}_{TWF}(\mathbb{M}, \mathbb{H}, \mathbb{R})$	**0.446**
$\text{GTB}_{TWF}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding VSR [11]	0.412
$\text{GTB}_{TWF}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding Collaborative Filtering (PMF) [15]	0.390
$\text{GTB}_{TWF}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding Content-based Filtering (LDA) [3]	0.386
$\text{GTB}_{TWF}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding TWF [10]	0.338

Table 4. Recall@50 obtained by controlled ablation testing using sufficient **version** information.

Feature	**Recall@50**
$\text{GTB}_{VSR}(\mathbb{M}, \mathbb{H}, \mathbb{R})$	**0.418**
$\text{GTB}_{VSR}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding TWF [10]	0.396
$\text{GTB}_{VSR}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding Collaborative Filtering (PMF) [15]	0.361
$\text{GTB}_{VSR}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding VSR [11]	0.344
$\text{GTB}_{VSR}(\mathbb{M}, \mathbb{H}, \mathbb{R})$, excluding Content-based Filtering (LDA) [3]	0.335

- Similar to our ablation testing with TWF in Section 5.3.2, under a dataset with full version information, we observe a reordering of recommendation techniques.
- Even though VSR does not displace collaborative filtering in this ablation testing, it still results in the second largest dip in recall scores (0.344) when we ablate it from the unifying model. In addition, under this dataset, improvement in recall obtained by VSR increases from 16.5% (in Table 2) to 22%.
- This further substantiates that when certain evidence is accessible, it changes the way signals are used in the unifying model, which the reordering of recommendation techniques in our ablation study suggests.

The ablation studies on the two controlled datasets (pertaining to full Twitter and version information) clearly demonstrate the importance of TWF and VSR in app recommendation, without which we would not have been able to capture Twitter and version signals for the purpose of improving recommendation quality.

5.4 Feature Importance in GTB

We further analyze each component of the feature set in Figure 1 of the GTB($\mathbb{M}, \mathbb{H}, \mathbb{R}$) model based on the relative influence. GTB allows us to measure the importance of each component feature. Basically, the more often a feature is used in the split points of a tree, the more important the feature is. Feature importance is essential because the input features are seldom equally relevant. While only a few of them often have substantial influence on the response, the vast majority are irrelevant and could just as well have not been included. Thus, it is helpful to learn the relative importance or contribution of each input feature in predicting the response. Figure 3 shows the relative importances of the top features and gives the following insights (starting with the most important feature):

- Not surprisingly, the *average rating (all versions)* is the most important factor as, when the average rating is high, it is natural for users to download the app because of its positive ratings. Therefore, this feature can be used as a strong signal in the unifying framework to make a split in the decision tree. This reasoning is also similar for the *average rating (current version).*

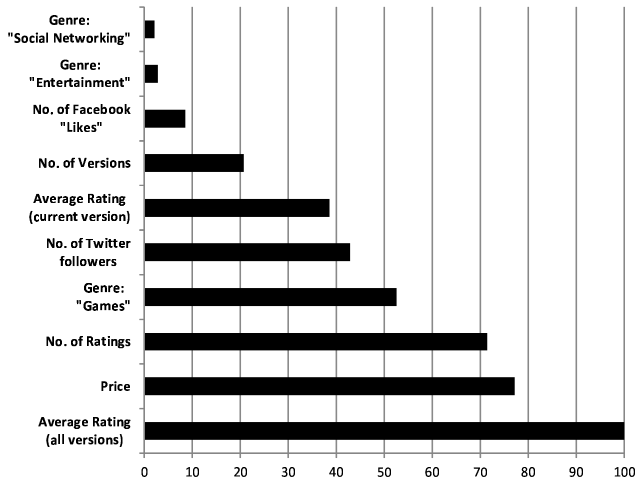

Fig. 3. Top features with the highest relative influence.

- *Price* (*i.e.*, free vs paid) is also an important factor, and this evidence coincides with the trend that apps in the app store are heading towards the freemium model — with the proportion of free apps taking up 90% of the app store. Therefore, the price of an app could be a strong signal for a split in the decision tree.
- The *number of ratings* is also a strong indicator, as the more ratings an app has garnered, the clearer the sign that it is popular and hence, likely to be consumed. It is also a clear sign that the collaborative filtering technique can be employed.
- Not only the *number of Twitter followers* to the app's Twitter handle is an indicator of a strong social reach, but also the availability of additional Twitter-followers information is an indicator that our Twitter-followers based recommendation technique can be utilized. Additionally, on a related note, the same reasoning could be used to explain why the *number of Facebook likes* is also one of the top features, as this indicator from Facebook is also a hint of the app's social presence on the popular social networking site.
- The *number of versions* also plays an important role as this is a sign that our version-sensitive recommendation technique (VSR) may be employed. Given that this feature is one of the top features of GTB, it suggests that the version-sensitive recommendation technique [11] is useful here.
- We also observe that some app genres fall under the top features, notably "games," "entertainment," and "social networking" — with "games" having a much more significant influence score. The three genres are consistent with alternate findings by Flurry Analytics[9] whereby they discovered that people spend most of their time in apps in the "games," "social networking," and "entertainment" genres across iOS and Android devices.

Finally, we also observe that our results of the top GTB features in Figure 3 coincide with another set of findings from Flurry Analytics, ComScore, and NetMarketShare[10].

[9] Flurry Analytics; "iOS & Android Smart Device Time Spent per App Category"; http://cl.ly/image/3m0P0g2r3f2C

[10] Flurry Analytics, ComScore, NetMarketShare; "Time Spent on iOS and Android Connected Devices"; http://cl.ly/image/201x2H1Q1j3H

For instance, the significant chunks that relate to genres (*i.e.*, "game," "entertainment," and "social messaging") coincide with our genre labels shown in Figure 3. Additionally, the "Facebook" and "Twitter" chunks also coincide with the "# of Facebook likes" and "# of Twitter followers" features in Figure 3, which suggests that apps that have a strong presence on these two popular social networks have a tendency to be spotted and subsequently consumed, making them popular candidates to be recommended. The data from the alternate user studies[11,12] demonstrates a strong correlation with our GTB feature component analysis shown in Figure 3. It indicates how two disciplines (*i.e.*, user studies and GTB feature component analysis) from two different sources of opinions and quantitive angles managed to arrive at similar findings. This further suggests a future direction in mobile app recommendation whereby more focus could be placed in user and trend analysis through social networks — a direction that deviates from traditional research in recommender systems.

6 Conclusion

Given that different recommendation techniques work in different settings, we evaluate a method for integrating the various sources of information into a hybrid model that can recommend a set of apps to a target user. We propose incorporating the user's prior history, app metadata, and the recommendation scores of various individual recommendation techniques into a hybrid recommendation model for app recommendation. We then use gradient tree boosting (GTB) as the core of the unifying framework to integrate the recommendation scores by using user features and app metadata as additional features for the decision tree. Experimental results show that the unifying framework achieves the best performance against individual and hybrid baselines. We also performed a series of in-depth analysis through ablation studies, and demonstrated how different pieces of evidences (such as Twitter and version information) that, when available, could be utilized sufficiently, and how the unifying model dynamically alters the recommendation based on available signals. Finally, we discovered an interesting correlation between important feature components in our unifying framework and user analysis from third-party data analytics companies, which further suggests a future direction in mobile app recommendation, where more focus could be placed in user and trend analysis via social networks.

References

1. R. Baeza-Yates, D. Jiang, F. Silvestri, and B. Harrison. Predicting The Next App That You Are Going To Use. In *Proc. of the 8th ACM International Conference on Web Search and Data Mining (WSDM'15)*, pages 285–294, 2015.
2. U. Bhandari, K. Sugiyama, A. Datta, and R. Jindal. Serendipitous Recommendation for Mobile Apps Using Item-Item Similarity Graph. In *Proc. of the 9th Asia Information Retrieval Societies Conference (AIRS 2013)*, pages 440–451, 2013.
3. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.
4. N. Chen, S. C. H. Hoi, S. Li, and X. Xiao. Mobile App Tagging. In *Proc. of the 9th ACM International Conference on Web Search and Data Mining (WSDM'16)*, pages 63–72, 2016.

[11] Flurry Analytics; "iOS & Android Smart Device Time Spent per App Category"; http://cl.ly/image/3m0P0g2r3f2C

[12] Flurry Analytics, ComScore, NetMarketShare; "Time Spent on iOS and Android Connected Devices"; http://cl.ly/image/201x2H1Q1j3H

5. N. Chen, S. C. H. Hoi, and X. Xiao. SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning. In *Proc. of the 8th ACM International Conference on Web Search and Data Mining (WSDM'15)*, pages 305–314, 2015.
6. E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López. Which App? A Recommender System of Applications in Markets: Implementation of the Service for Monitoring Users' Interaction. *Expert Systems with Applications*, 39(10):pages 9367–9375, 2012.
7. C. Davidsson and S. Moritz. Utilizing Implicit Feedback and Context to Recommend Mobile Applications from First Use. In *Proc. of the 2011 Workshop on Context-awareness in Retrieval and Recommendation (CaRR'11)*, pages 19–22, 2011.
8. J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29:1189–1232, 2001.
9. J. Lin. *Mobile App Recommendation*. PhD thesis, National University of Singapore, 2014.
10. J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua. Addressing Cold-Start in App Recommendation: Latent User Models Constructed from Twitter Followers. In *Proc. of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'13)*, pages 283–292, 2013.
11. J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua. New and Improved: Modeling Versions to Improve App Recommendation. In *Proc. of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'14)*, pages 647–656, 2014.
12. 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 *Proc. of the 8th ACM International Conference on Web Search and Data Mining (WSDM'15)*, pages 315–324, 2015.
13. W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A Survey of App Store Analysis for Software Engineering. Technical Report RN/16/02, University College London, 2016.
14. D. H. Park, M. Liu, C. Zhai, and H. Wang. Leveraging User Reviews to Improve Accuracy for Mobile App Retrieval. In *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*, pages 533–542, 2015.
15. R. Salakhutdinov and A. Mnih. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *Proc. of the 25th International Conference on Machine Learning (ICML'08)*, pages 880–887, 2008.
16. C. Wang and D. M. Blei. Collaborative Topic Modeling for Recommending Scientific Articles. In *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pages 448–456, 2011.
17. J. Wang, Y. Zhang, and T. Chen. Unified Recommendation and Search in E-Commerce. In *Proc. of the 8th Asia Information Retrieval Societies Conference (AIRS 2012)*, pages 296–305, 2012.
18. Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying Diverse Usage Behaviors of Smartphone Apps. In *Proc. of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC'11)*, pages 329–344, 2011.
19. B. Yan and G. Chen. AppJoy: Personalized Mobile Application Discovery. In *Proc. of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*, pages 113–126, 2011.
20. P. Yin, P. Luo, W.-C. Lee, and M. Wang. App Recommendation: A Contest between Satisfaction and Temptation. In *Proc. of the 6th International Conference on Web Search and Data Mining (WSDM'13)*, pages 395–404, 2013.
21. A. Zhang, A. Goyal, R. Baeza-Yates, Y. Chang, J. Han, C. A. Gunter, and H. Deng. Towards Mobile Query Auto-Completion: An Efficient Mobile Application-Aware Approach. In *Proc. of the 25th International World Wide Web Conference (WWWW 2016)*, pages 579–590, 2016.
22. V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach. In *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI'10)*, pages 236–241, 2010.
23. H. Zhu, H. Xiong, Y. Ge, and E. Chen. Mobile App Recommendations with Security and Privacy Awareness. In *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*, pages 951–960, 2014.