Climbing the App Wall: Enabling Mobile App Discovery through Context-Aware Recommendations

Alexandros Karatzoglou
Telefonica Research
Barcelona, Spain
alexk@tid.es

Linas Baltrunas
Telefonica Research
Barcelona, Spain
linas@tid.es

Karen Church
Telefonica Research
Barcelona, Spain
karen@tid.es

Matthias Böhmer
German Research Center for Artificial Intelligence
Saarbrücken, Germany
matthias.boehmer@dfki.de

ABSTRACT

The explosive growth of the mobile application (app) market has made it difficult for users to find the most interesting and relevant apps from the hundreds of thousands that exist today. Context is key in the mobile space and so too are proactive services that ease user input and facilitate effective interaction. We believe that to enable truly novel mobile app recommendation and discovery, we need to support real context-aware recommendation that utilizes the diverse range of implicit mobile data available in a fast and scalable manner. In this paper we introduce the *Djinn* model, a novel context-aware collaborative filtering algorithm for implicit feedback data that is based on tensor factorization. We evaluate our approach using a dataset from an Android mobile app recommendation service called *appazaar*[1]. Our results show that our approach compares favorably with state-of-the-art collaborative filtering methods.

Categories and Subject Descriptors

H3.3 [**Information Search and Retrieval**]: Information filtering—*Collaborative Filtering*; G3 [**Probability and Statistics**]: Correlation and regression analysis; G1.6 [**Optimization**]: Gradient methods

Keywords

Collaborative Filtering, Tensor Factorization, Context, Implicit Feedback, Mobile apps, Mobile Recommendation

[1]See: `http://appazaar.net` last accessed Nov 2011

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

1. INTRODUCTION

According to *AppsFire*, there are currently 1 million mobile apps available between Apple's AppStore and the Android Market[2]. However, this explosive growth has lead to a new challenge facing mobile users. That is, finding the most interesting and relevant apps from the hundreds of thousands that exist. A number of industry solutions have emerged that provide app recommendation and aggregation services which attempt to filter, rank and recommend the best apps to end users. Most of these services enable sharing and discovery of mobile apps through (1) ratings, which suffer from tedious explicit input from users, and (2) simplistic recommendations based on: users profiles, the apps installed by the end-user, and in some cases the duration/usage of those apps. However, few of these services use any form of mobile context information (e.g. location, time, etc.) to facilitate the recommendations and most of these services base their recommendations on explicit input (i.e. ratings) or weak implicit data (i.e. an example of weak indicator of whether the user is interested in an app is that the user downloaded the app).

Recent research has shown that key mobile contexts like location, time, activity and social interactions have a significant impact on the information needs and behaviours of mobile users [4]. The same is true for mobile app usage [1].

In this paper we introduce the *Djinn* model: a novel context-aware collaborative filtering method for implicit data that is based on tensor factorization. We deal with the dual challenge of building a preference model in the absence of explicit feedback information from the user and incorporating real contextual information into a single recommendation model. Note that our method can be used in any context-aware recommendation setting with implicit feedback data.

2. RELATED WORK

Context Aware Recommender Systems.

Context-Aware Recommender Systems (CARS) are based on the fact that contextual factors (such as *e.g.* time, location, activity, weather, emotional state, social surrounding) have a heavy influence on the recommendation needs of users. In [8] a context-aware collaborative filtering model for explicit data (i.e. ratings) that is based on tensor factor-

[2]See: `http://tcrn.ch/k9k3Ro` last accessed Nov 2011

ization was introduced . However, as mentioned previously CF models for explicit data are not adequate for the case of implicit data. Moreover we show analytically how our new model scales favorably when compared to [8].

Implicit Feedback data.

In implicit feedback the numerical values of the interaction between items and users (counts/number of clicks etc.) can only be considered as an indirect indication or a confidence measure of how much the user liked an item. Moreover for non-observed $user - item$ interactions we cannot be certain as to whether the user did not consider the items or if the user considered the items and simply chose not to interact with the items (reflecting a negative feedback). Hence we cannot ignore these entries as this would lead to a recommender that would be overly optimistic with regard to user preferences.

The matrix factorization approach for implicit data introduced in [7] uses a trick that exploits the sparse structure of the data (dominated by non-observed entries) to speed up the optimization process. Here we introduce a generalized version of this *trick* applied to N-dimensional tensor factorization for the optimization of the *Djinn* model. Implicit feedback factor models were also introduced in [9] and [10], where once again the sparsity of the data was exploited in order to speed up optimization. However, none of these implicit feedback based methods utilize any form of context information.

Mobile Application Recommendation.

Woerndl et al. [11] explores the use of location to recommend mobile apps. Their system works by recommending apps that others users have installed or used frequently at a given location. *AppJoy* [12] supports personalized mobile app recommendations by combining item-based collaborating filtering with data on how the user actually uses his/her installed apps. More recently, a wide range of commercial services have emerged. For example, *Appolicious* ranks and recommends apps using reviews, friend recommendations and the apps currently installed by the end user. *AppsFire* and *Zwapp* support discovery of apps among friends. *Smokin Apps* and *AppBrain* for Android match users with apps they would like based on apps similar to those the user has currently installed. Finally, *AppSpot* facilitates recommendation based on the apps you already have, as well as apps you have searched for in the past. Overall existing research has relied heavily on explicit user ratings, recommendations from friends or weak implicit data such as app installation.

3. CONTEXT-AWARE RECOMMENDATIONS

3.1 Tensor Factorization (TF)

N-dimensional TF extends CF methods to N-dimensional data where the additional dimensions represent the context of the recommendation. In the example of figure 1, we depict the case of mobile app recommendation with *location* as an additional context variable. In this example three users $\{u_1 = \text{Mary}, u_2 = \text{John}, u_3 = \text{Elen}\}$ interact with the apps $\{m_1 = \text{Barcode}, m_2 = \text{Task Killer}, m_3 = \text{Angry Birds}, m_4 = \text{Facebook}\}$ in context *location* $\{c_1 = \text{Work}, c_2 = \text{Home}, c_3 = \text{Car}\}$. The resulting tensor $Y \in \mathbb{N}^{n \times r \times l}$, where n is the number of users, r the number of items and l the number of context categories, contains the frequency (represented by integer values) of interaction between the users, apps and the location i.e the number of times a user interacted with an app at a particular location. An absence of an interaction is signaled with a 0.

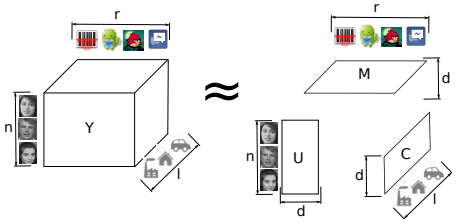

Figure 1: Illustration of the (3-dimensional) tensor factorization *Djinn* model.

The main idea of the *Djinn* model's algorithm is to decompose this tensor into three matrices $U \in \mathbb{R}^{n \times d}$, $M \in \mathbb{R}^{r \times d}$ and $C \in \mathbb{R}^{l \times d}$ (see figure 1), using the Candecomp-Parafac (CP) model [6] which corresponds to learned factors that form a profile for each user, app, and context variable respectively. We denote by U_{i*} the entries of the i_{th} row of matrix U. The decision function that would provide a score for the recommendation engine for a *user* u_i, *item* m_j, *context* c_k is given by

$$F_{ijk} = \langle U_{i*}, M_{j*}, C_{k*} \rangle = \sum_{g=1}^{d} U_{ig} M_{jg} C_{kg} \qquad (1)$$

For the sake of simplicity, we will describe the model for a single contextual variable C. In fact, our method allows for an arbitrary number of context variables to be included into the model.

3.1.1 Model

In the *Djinn* model observed $user - item - context$ interactions p_{ijk} are signalled with a 1 and unobserved with a 0. Y_{ijk} are the interaction counts between the user u_i, item m_j under context c_k as a user often interacts several times with the same item, e.g. clicking on a news article or using an app. Our confidence in the 0 values is low as these values could represent negative feedback or they could indicate that the user simply did not consider the item.

We take these counts to reflect the confidence we have in the feedback from the user i.e. an item that is repeatedly used/consumed should reflect a greater confidence in the preferences of the user over an item that is used/consumed only once. We transform the counts of usage of an item to "confidence" by introducing a new variable w_{ijk} given by:

$$w_{ijk} = \begin{cases} \alpha log(1 + Y_{ijk}) + log(1 + \frac{m}{m^{u_i}+1}) & Y_{ijk} > 0 \\ 1 & Y_{ijk} = 0 \end{cases} \qquad (2)$$

where m^{u_i} is the number of items used by user u_i and α a parameter which we set to 0.8 for all our experiments. The first term in eq. 2 reflects the confidence regarding items that are often consumed while the second term reflects the fact that confidence in items should be high only if very few items are used.

The aim of the model is to compute the factors for the user $U^{n \times d}$, item $M^{r \times d}$ and context $C^{l \times d}$ matrices using historical consumption data. Once the factors have been calculated a prediction can be computed using eq. 1. In the proposed TF

model we compute these factors by minimizing the following objective function:

$$\min_{U,M,C} \sum_{i}^{n}\sum_{j}^{r}\sum_{k}^{l}[w_{ijk}\left(p_{ijk} - \langle U_{i*}M_{j*}C_{k*}\rangle\right)^2 + \frac{\lambda}{n}||U_{i*}||^2 + \frac{\lambda}{r}||M_{j*}||^2 + \frac{\lambda}{l}||C_{k*}||^2] \quad (3)$$

The term $\frac{\lambda}{n}||U_{i*}||^2 + \frac{\lambda}{r}||M_{j*}||^2 + \frac{\lambda}{l}||C_{k*}||^2$ is required for regularization.

3.1.2 Optimization

We optimize the objective function for the *Djinn* model (eq. 3) using Alternating Least Squares (ALS). When optimizing over a single factor matrix e.g. U while keeping the remaining factor matrices fixed, we observe that the cost function becomes quadratic and there is an analytic solution. We illustrate how to optimize the objective with respect to the user factor matrix U. We differentiate the objective function and set the derivative to zero. Solving with respect to a single user u_i factor vector gives:

$$U_{i*} = \left(\sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T w_{ijk}\left[M_{j*}\odot C_{k*}\right] + \frac{\lambda}{n}I\right)^{-1} \times \sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T w_{ijk}p_{ijk} \quad (4)$$

where $\odot$ is the Hadamar or element-wise product and I the identity matrix of size d. Directly computing this expression would scale $O((nl)^2d)$ which would be prohibitively expensive even for relatively small datasets. We can achieve a very significant speedup by rewriting this expression as:

$$\sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T w_{ijk}\left[M_{j*}\odot C_{k*}\right] = \sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T\left[M_{j*}\odot C_{k*}\right] + \sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T\left(w_{ijk}-1\right)\left[M_{j*}\odot C_{k*}\right] \quad (5)$$

The first part $\sum_j^r\sum_k^l\left[M_{j*}\odot C_{k*}\right]^T\left[M_{j*}\odot C_{k*}\right]$ is now independent of the user and can be precomputed at the beginning of the iteration over each user factor matrix. In the second part $\sum_j^r\sum_k^l\left[M_{j*}\odot C_{k*}\right]^T\left(w_{ijk}-1\right)\left[M_{j*}\odot C_{k*}\right]$ the expression $(w_{ijk}-1)$ is 0 for all the non-observed values in the tensor Y, which are the vast majority of entries in the tensor, and thus we can compute it over the non-zero values S_i^{u+} of user u_i in the tensor Y. This vastly improves computation time and scalability to $O((d)^2n^{u_i+}+d^3)$ where n^{u_i+} the number of positive feedback items for user u_i and assuming cubic scalability for the matrix inversion. We can thus rewrite this expression as:

$$\sum_{j}^{r}\sum_{k}^{l}\left[M_{j*}\odot C_{k*}\right]^T\left[M_{j*}\odot C_{k*}\right] + \sum_{j,k\in S_{i+}^u}\left[M_{j*}\odot C_{k*}\right]^T\left(w_{ijk}-1\right)\left[M_{j*}\odot C_{k*}\right] \quad (6)$$

Since we need to compute this over each user the scalability becomes $O(d^2K+d^3n)$. We speed up the computations using the fact that: $\sum_j^r\sum_k^l\left[M_{j*}\odot C_{k*}\right]^T\left[M_{j*}\odot C_{k*}\right] = M^TM\odot C^TC$ since matrix libraries handle these type of operations (matrix multiplications and element-wise products) very efficiently. We can compute the item M and context factors C in a similar manner.

The optimization procedure is repeated over each factor matrix until convergence. The whole algorithm scales $O(d^2K + d^3(n+r+l))$ i.e linearly to the number of observed entries K and usually $K \gg (n+r+l)$.

4. EXPERIMENTS

Appazaar data.

Appazaar recommends mobile apps to its users from the Android Market. Along with tracing mobile app usage, *appazaar* also tracks available context information from the phones sensors. This context information is currently not used by *appazaar* to facilitate context-aware recommendations [2]. In total, the *appazaar* dataset contains 3,260 users, 18,205 items and 3.7 million records about the usage of apps. Apart from the user and app id's the features that can be extracted are:

- **Moving**: Whether the user was moving with walking speed (3), faster (4) or standing still (2); or this information is not available (1).
- **Location**: A heuristic describing whether the user is at home, at work or elsewhere. We have set the most frequent place from 6am to 6pm as work (1), the most frequented place from 6pm to 6am as home (2), and defined all other locations as elsewhere (3).
- **Time of day**: The time of the day in blocks of 2 hours, from 12pm-2am (1) to 10pm-12pm (12).
- **Day of the week**: From Sunday (1) to Saturday (7).
- **Number of times used**: The number of times the app was used by the user with regard to the other parameters.

Note that we have 4 contextual factors plus user and item dimensions thus we model the data with a 6-dimensional tensor.

Evaluation Protocol.

We temporally order the data and split it with the first 80% of the data forming the training set and the remaining 20% the test set. The data was then aggregated for each contextual combination found. For example, user u_i used application m_j while being still at home, between 6pm to 6am on a Weekend 25 times. In order to facilitate the comparison to non-context aware methods we filtered the test set so that for each $user-item$ combination, only one context combination is included in the test set.

For the testing procedure we adopt a similar strategy to [5]. We first randomly select 1000 additional apps that the user did not use. We assume that these items are not of interest to user u_i. We predict the scores for the test items for user u_i in an associated context and for the additional 1000 items. We form a ranked list by ordering all the items according to their predicted scores. This lists over all users are then used to produce Precision-Recall plots. Moreover, we also compute the Mean Average Precision evaluation measure. . We used validation set to tune the parameters of the model.

We conduct 5 runs for each experimental setup and average the results. We do not report on the variance of the results since it was insignificant in our experiments. Moreover, all differences between *Djinn* and the other methods are statistically significant with $p - value < 0.001$.

Methods in Comparison.

First, we compare our results against a standard regularized matrix factorization method (denoted *MF*) (based on Simon Funk's[3] approach) where the non-observed 0 values are ignored and a regression is performed on the p_{ijk} values derived from eq. 2. Naturally, *MF* also ignores the contextual information. We also implemented a method proposed in [7] which is essentially matrix factorization for implicit feedback data (denoted *iMF*). *iMF* takes into account only users and items and ignores additional context but treats the 0 values of the non-observed $user - item$ pairs as negative feedback. Moreover we implemented a Poisson regression (denoted *Poisson*) based on [3] where contextual information is used as a feature vector $\mathbf{x}$ to predict the usage counts y_i for each item m_j. As a baseline we used the overall popularity of the apps for recommending items to the users (denoted *AVG*).

4.1 Experimental Results

We first compute the MAP of all the methods after tuning their parameters, the summary of the results is shown in Table 1. *Djinn* improves *MAP* over the non-context aware method, *iMF*, by 28%. This clearly indicates significant benefits of using context in the mobile domain for app recommendations. Interestingly, *MF* which is currently considered state-of-the-art in rating-based datasets shows poor performance. This is in line with the findings of [5]. The poor performance of non-personalized methods such as *AVG* or *Poisson* indicates the huge differences of app usage between users and the need for personalization.

Table 1: MAP of the methods on *appazaar* data.

AVG	*Poisson*	*MF*	*iMF*	*Djinn*
0.046	0.072	0.201	0.326	**0.410**

The precision-recall plots for all methods, on the *appazaar* data are shown in Figure 2. Our results are consistent with the previous experiment, that is, *Djinn* outperforms all other methods and that the addition of context has a substantial impact on performance. The most interesting results can be found on the left side of the graph which shows precision. In mobile recommendation precision is more important than recall given the need for recommending a short list of the most relevant apps. We again observe that methods that take into account negative feedback (*Djinn*, *iMF*) in the form of the non-observed entries outperform methods that ignore this feedback (*MF*).

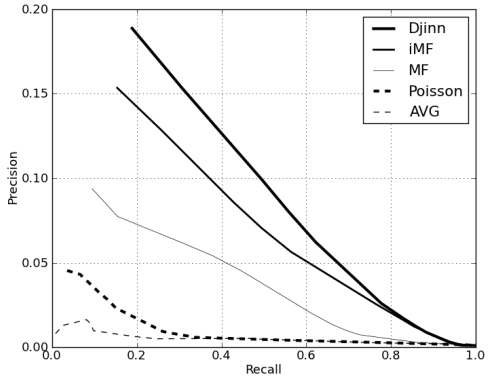

Figure 2: Precision/recall plots for *appazaar* data. *Djinn* outperforms all other methods and methods that take into account negative feedback perform better then those that do not.

5. ACKNOWLEDGMENTS

This work is funded as part of a Marie Curie Intra European Fellowship for Career Development (IEF) awards held by Alexandros Karatzoglou (CARS, PIEF- GA-2010-273739).

[3] `http://sifter.org/~simon/journal/20061211.html`

6. REFERENCES

[1] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with angry birds, facebook and kindle - a large scale study on mobile application usage. In *Proc. of Mobile HCI '11)*. ACM, 8 2011.

[2] M. Böhmer, M. Prinz, and G. Bauer. Contextualizing mobile applications for context-aware recommendation. In *Adj. Proc. of Pervasive 2010*, 2010.

[3] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 209–218, New York, NY, USA, 2009. ACM.

[4] K. Church and B. Smyth. Understanding the intent behind mobile information needs. In *Proceedings of IUI '09*, pages 247–256. ACM, 2009.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. of RecSys '10*, pages 39–46, 2010.

[6] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Math. and Physics*, 6, 1927.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. of ICDM '08*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[8] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proc. of RecSys '10*, pages 79–86, 2010.

[9] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proc. of KDD '09*, pages 667–676, New York, NY, USA, 2009. ACM.

[10] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1055–1060, Washington, DC, USA, 2010. IEEE Computer Society.

[11] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *Proceedings of the IEEE ICDE*, 2007.

[12] B. Yan and G. Chen. Appjoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 113–126. ACM, 2011.