Improving Augmented Reality Using Recommender Systems

Zhuo Zhang, Shang Shang, Sanjeev R. Kulkarni
Department of Electrical Engineering
Princeton University
Princeton, NJ
USA
{zhuoz,sshang,kulkarni}@princeton.edu

Pan Hui
CSE Department
The Hong Kong University of Science and Technology
Hong Kong
panhui@cse.ust.hk

ABSTRACT

With the rapid development of smart devices and wireless communication, especially with the pre-launch of Google Glass, augmented reality (AR) has received enormous attention recently. AR adds virtual objects into a user's real-world environment enabling live interaction in three dimensions. Limited by the small display of AR devices, content selection is one of the key issues to improve user experience. In this paper, we present an aggregated random walk algorithm incorporating personal preferences, location information, and temporal information in a layered graph. By adaptively changing the graph edge weight and computing the rank score, the proposed AR recommender system predicts users' preferences and provides the most relevant recommendations with aggregated information.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Augmented Reality, Recommender System, Random Walk, PageRank, Graph, High-dimensional

1. INTRODUCTION

Augmented reality (AR) is an emerging technology that embraces cutting-edge developments in sensing and smart devices, wireless communication, pervasive computing and intelligent environments. It augments users' experiences by adding virtual objects and allowing users to interact in real time and space [4]. Current AR systems are mainly based on smart devices such as Google Glass and iPhone, with limited screen size. Content displayed on a small screen can be difficult and frustrating for end-users, thus high performance recommender systems are desired in AR systems [8]. Although recommendation algorithms, especially collaborative filtering techniques, have been intensively studied during the past decade, we cannot trivially adapt and deploy traditional recommender in AR systems, because they differ in the following respects:

- Location: Nearby recommendations are usually more interesting than a place in a remote location.
- Timing: Short-term or in-time preferences have high priority. For example, at Sunday 11am, a user is more likely looking for a restaurant for brunch rather than a night club.
- Cold start: New users are sensitive to application user experience but sparse data may lead to inaccurate recommendations.
- Immediate feedback: AR recommender systems have to react quickly based on users' behaviors. Users may click interesting items immediately after a recommendation is made. Recommendation lists are then updated accordingly.

Currently most AR applications use distance-based filtering and visibility-based filtering. Some advanced techniques include a spatial model, e.g., [6], which uses focus and nimbus to determine the importance of objects. Alternatively, model-based filtering with prior knowledge of human motivation has been applied in AR recommendation [9]. Popular AR applications such as Wikitude and Layar also use this recommendation. Individual preference can be captured quite well, though they fail to combine historical information and other users' data for the recommendation.

In this paper, we propose a random walk algorithm for AR recommendation, incorporating user preference, behavior patterns, history records, and social network information. We construct a layered graph consisting of users, places and labels as nodes, and we incorporate temporal and location information into the edge weight. We simulate the user decision making process as a biased random walk on the graph. The stationary distribution of the random walk represents the ranking score, inspired by the PageRank algorithm[7]. The rest of the paper is organized as follows: We formulate the AR recommendation problem in Section 2. We propose the random walk based algorithm in Section 3, followed by preliminary experimental results in Section 4. Conclusion and further work are discussed in Section 5.

*This research was supported in part by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370, and by support from Deutsche Telekom T-Labs.

RecSys'13, October 12–16, 2013, Hong Kong, China.

http://dx.doi.org/10.1145/2507157.2507211.

2. PROBLEM FORMULATION

In recommender systems, there are lists of both users and items. Each user provides scores or linguistic terms such as "like" or "dislike" to rate a subset of all possible items. Recommender systems such as Netflix and Amazon are designed to help select useful and relevant information for a specific user. It can be formulated as a matrix completion problem for the user-item rating matrix [2]. However, in some cases, extra information may help improve recommendation quality, such as temporal information, location information and social network information [12, 10]. One possible approach for incorporating this additional information is multi-dimensional collaborative filtering. [11] and [1] used a reduction-based algorithm and applied classic 2D collaborative filtering algorithms to produce a final recommendation. This approach has limitations. For example, when we extend available ratings into high dimension, the data will become extremely sparse and many existing algorithms will lose their effectiveness. Also, when multiple dimensions are decoupled as pairs of dimensions, relationships among more than two dimensions are likely to be lost. Therefore in this paper, we propose a layered graph to aggregate high dimensional information and use a random walk algorithm to make recommendations. The problem is formally formulated as below.

Assume that there is a set of users $U = \{u_1, ..., u_m\}$ and a set of places $I = \{i_1, ..., i_n\}$. Traditionally, a two dimensional rating matrix $U \times I \to R$ can be constructed. Each element $r_{u,i}$ in R denotes user u's rating of place i. The ratings can be either explicit, for example, on a 1-5 scale as in TripAdvisor, or implicit such as "visited" or "not visited". The rating data typically specifies only a small number of the elements of R. We further assume binary label and user social information is given. Let $L = \{l_1, ..., l_k\}$ be the set of label information of items. For example, for places, L can be restaurants, shopping malls, bus stops, etc. $L_i \in \{0, 1\}^k$ denote the features of place i, where k is the total number of labels. Correspondingly, let $S = (U, \varepsilon)$ contain social network information, represented by an undirected or directed graph, where U is a set of nodes while ε is a set of edges. $\forall u, v \in U$, $(u, v) \in \varepsilon$ if v is a friend of u. We further denote set of times as T and location set as P. Then the multidimensional rating matrix will be formulated as $R = U \times I \times L \times T \times P$. Given the target user u, the current time t and current location p, our ultimate goal is to find the optimal place defined below.

$$\forall u \in U, t \in T, p \in P, i_{u,t,p} = \arg\max_i R(u, l, i, t, p) \qquad (1)$$

In the following part, we will address the questions mentioned above with a proposed random walk algorithm.

3. RANDOM WALK IN AR RECOMMENDER SYSTEMS

Several random walk algorithms [12, 10] have been proposed for the traditional recommender systems but they fail to handle with the characteristics of AR systems. In this section, we will describe our random walk algorithm in detail, and discuss how to deal with specific issues in multidimensional AR recommender systems.

3.1 Model Construction

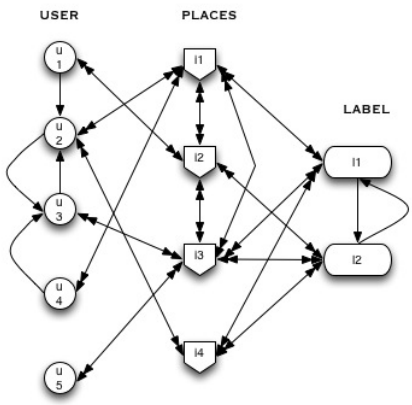

Figure 1: An example of recommendation graph

3.1.1 Graph Formulation

Let $G = \{V, E\}$ be a directed graph model for AR recommender systems, as shown in Fig.1. We construct the graph by the following rules. 1) Nodes V represent constant attributes such as users, places and labels; for each entity all the nodes stay in the same layer; 2) Edges/Weighted edges E represent variables such as locations or time, or relationships, e.g. social network information.

In this recommender system, the nodes $V = U \cup I \cup L$ form three layers, which consist of users, places and labels. The edges E are classified as the following five classes based on the layers that the nodes belong to. Higher weight means higher chance to transition from one node to another. We incorporate personal records, location information, and label information into the graph. Note that we use the inverse exponential distribution to model a human's mobility [3]. Let d denote the distance between current location and the target place. Then the human's mobility is modeled by the distribution $\frac{1}{Z}\exp(-\alpha d)$ where $\alpha \geq 0$ is a decay parameter and Z is a normalization factor. α is a tunable factor set by users. For example, if $\alpha = 0$, distance to the current location would not affect recommendation results. In contrast, if α is large, only nearby places will be recommended. Experimental results in Section 4 show the effect of α on average distance between local position and recommendation.

- For $u \in U, i \in I, (u, i) \in E$ and $(i, u) \in E$ if and only if user u has visited i (assume only implicit ratings are available) and the weight $w_{u,i} = \exp(-\alpha(d(i, p)), w_{i,u} = 1$ where $d(:, :)$ is the distance between two places, p is the current location.

- For $i \in I, l \in L, (i, l) \in E$ and $(l, i) \in E$ if and only if $L_i^l \neq 0$, i.e., the place i belongs to label l and the weight $w_{i,l} = w_{l,i} = 1$.

- For $u_1, u_2 \in U, (u_1, u_2) \in E$ if and only if $(u_1, u_2) \in \varepsilon$, which means u_2 is a friend of u_1. Note that the relationship in social networks is not necessarily mutual such as "follow" in Twitter.

- For $i_1, i_2 \in I, (i_1, i_2) \in E$ and $(i_2, i_1) \in E$ if and only if $i_1 \neq i_2$. Define $w_{i_1,i_2} = \exp(-\alpha(d(i_1, i_2))$.

- For $l_1, l_2 \in L, (l_1, l_2) \in E$ if and only if the transition probability from label l_1 to l_2 is greater than 0, which we will get from the training data set.

3.1.2 Transition Probability

Assume that a random walk with an initial distribution is applied in such a weighted graph. The path is a sequence of random variables $X_1, ..X_t, ...$, which form a Markov chain and the future state only depends on the current state. We need to further normalize the weight to make it a transition probability.

Let $Y_1, Y_2, ..., Y_s$ denote s layers in the graph. We first define the transition probability T_{ij} between different layers Y_i, Y_j.

$$T_{ij} := \sum_{n_1 \in Y_i, n_2 \in Y_j} P(X_{t+1} = n_2 | X_t = n_1). \quad (2)$$

Specifically in this case, we have 3 layers U, I, L and we define $T_{ij} = \frac{1}{3}, \forall i, j \in \{1, 2, 3\}$.

We further define the transition probability between different nodes $n_i \in Y_x, n_j \in Y_y$. It is normalized by all weights to the layer Y_y times a layer transition probability T_{xy}.

$$P_{ij} := P(X_{t+1} = n_j | X_t = n_i) = \frac{w_{n_i,n_j} T_{xy}}{\sum_{n \in Out_i \cap n \in Y_y} w_{n_i,n}}, \quad (3)$$

where $Out_i = \{n | (n_i, n_j) \in E\}$.

3.1.3 Temporal Information

Another important factor in the AR recommender system is the temporal information. For example, users at noon may look for restaurants rather than nightclubs, while at around 3pm they might prefer coffee or ice cream rather than fine food. Accordingly, we will calculate the probability $P_u(t)$ of every label activities within each time slot, say half an hour or an hour. Here $P_u(t)$ is a $k \times 1$ histogram distribution vector to denote the probability that a specific user u looks for some places related to label l at time slot t.

3.2 Score Computation

3.2.1 Random Walk

For the recommendation graph $G = (V, E)$, let the $|V| \times 1$ vector θ denote the customized probability vector. We will illustrate how to set θ based on a customized request in the following. We define another parameter $\beta \in [0, 1]$, called the damping factor. With probability β, the random walk will continue its path in G. Otherwise, it will go back to the customized probability distribution θ. Let the $|V| \times |V|$ matrix M denote the Markov transition matrix, in which $M_{ij} = P_{ji}$ in Eqn.(3). We further define γ as the stationary distribution for this random process. It satisfies the following equations:

$$\gamma = \beta M \gamma + (1 - \beta)\theta \quad (4)$$

Therefore, we can transform Eqn.(4) into

$$\gamma = (\beta M + (1 - \beta)\theta 1^T)\gamma, \quad (5)$$

where 1^T is a $|V| \times 1$ vector and $1^T \gamma = |\gamma|_1 = 1$.

We define $A = \beta M + (1-\beta)\theta 1^T$ and the rank score γ would be the eigenvector of A. In the following section we will assign suitable θ for different purposes. Based on Eqn.(5), we can calculate the rank score γ.

3.2.2 Temporal Information

For a specific time t and a target user u at a location p, we aggregate the user information and location information into the graph. In order to combine $P_u(t)$ and γ to make an effective recommendation, we let Q be the $|V| \times k$ label matrix where Q_{ij} denote whether v_i belongs to label l_j. $Q_{ij} = 1$ if and only if v_i is a place node (i.e., in the place layer) and v_i belongs to label l_j. Otherwise $Q_{ij} = 0$. Then our final recommendation score will be

$$\gamma(t) = \gamma \cdot (Q \times P_u(t)), \quad (6)$$

where $\cdot$ is the dot product between two vectors.

3.3 Personalized Recommendation through θ

Now the only challenge for this random walk algorithm is how to set θ to meet each user's personal requirement in AR systems.

3.3.1 Regular Case

We set $\theta = e_u$ where e_u is the $|V| \times 1$ unit vector, and get the rank score γ. We sort all the nodes in V based on γ and select the top n places for the top n recommendations.

3.3.2 Cold Start Case

The cold start problem has been one of the most important issues in recommender system for years. For AR recommender systems, it is crucial since new users will not tolerate a bad user experience for a long time. In other words, if the AR recommender systems cannot give good predictions for the first several attempts, the new user may quite possibly delete the application forever. However sparse data may lead to inaccurate personal rank score γ for new users. Averages based on relatively low support (small values of $|I_u|$, i.e. the number of places that user u has visited) can generally be improved by shrinkage towards a common mean $\bar{\gamma}$ [5]. Set $\theta = \frac{1}{|U|}\mathbb{1}_{\{v \in U\}}$ and we can compute the rank score $\bar{\gamma}$ for overall users. Therefore we can further define the rank score for cold start users as follows.

$$\gamma_{cold} = \frac{|I_u|\gamma + \tau\bar{\gamma}}{|I_u| + \tau}, \quad (7)$$

where the parameter τ controls the extent of the shrinkage.

3.3.3 Interaction/Update Case

Users will interact with an AR system in real time. When several selected places are shown in the AR system, users will use their finger to click on the item, denoted as i, which they are interested in. Here we propose two methods for further updates, listed below.

- Label-driven Update: This method is fast and focuses on the label behind the place i only. Every time user u selects place i among all the available items the recommender system provides, we regard u's purpose to be the label set to which i belongs. We then replace all the top n recommendations by the top n places that belongs to these labels.

- Place-driven Update: This method is relatively slow and needs to recompute the rank score. Basically, we will set $\theta = e_i$ to compute the rank score γ for the recommendations.

Table 1: Average Percentile of Recommendations

α	0	0.1	0.5	1
Percentile(%)	93.5	76.3	57.4	50.1

Table 2: Recall of Top Recommendations

Recall (%)	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1$
Top 10	83.3	32.1	26.6	24.5
Top 30	95.2	75.1	47.1	30.1
Top 50	100.0	81.0	65.3	43.0

Table 3: Average Distance of Top Recommendations

Distance (km)	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1$
Top 10	18.46	7.52	5.82	3.65
Top 30	20.01	9.30	7.63	6.34
Top 50	22.85	13.32	10.90	9.60

4. PRELIMINARY EXPERIMENTAL RESULT

We downloaded the Gowalla dataset[1], which contains 19,183 users, 30,367 places in NYC and 357,753 check-ins in each of which records a specific place a user has visited. We randomly select 100 users, 100 most popular places and 831 corresponding check-ins to construct the transition graph. Due to the incomplete data, in this experiment the layered graph only contains users, locations and check-in information. Then a target user is randomly picked and the current location is randomly generated. We use the proposed algorithm to recommend places to the target user and calculate three measures of the top recommendation list. *Percentile* is the average position (in percentage) of the actual visited places out of the whole set of places, where 99% denotes the top 1%. *Recall* is the number of hits (i.e., the visited places) of the top n recommendations divided by the total number of visited places. These two measures can evaluate the effectiveness of recommendations. The *average distance* of the top recommendations can evaluate whether the location factor is taken into account in the recommendations. In the experiment, we set $\beta = 0.85$ and vary α to see the different performance of the algorithm. Note that when $\alpha = 0$, the algorithm will become the traditional random walk without location information. We compare the average percentile, recall, and average distance with different α. We repeatedly generate target user and current location 500 times. The average results are reported in Table 1, 2 and 3.

The results show that incorporating location indeed improves AR applications by reducing the average distance of top recommendations. When α becomes larger, the average distance decreases while the percentile and recall decrease as well. Therefore the value of α can deal with the tradeoff between recommendation accuracy and average distance. Larger α means higher priority of distance and lower weight on personal tastes.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed using recommender systems to improve AR, specifically by applying a random walk algorithm to AR recommender systems. We incorporate location information into the weight of edge in the graph and incorporate temporal statistics as well into the rank score. Therefore we aggregate the user's personal preference, location information and temporal information into the biased random walk to recommend places for AR device users. We have also proposed the recommendation list updating methods when users take actions. Preliminary results show that location is indeed incorporated into the layered graph.

For future work, it may be interesting to find more comprehensive datasets with complete information on history record, location and temporal preference. Parameter tuning is another key issue to handle the tradeoff between different factors. Furthermore, user feedback can help provide a true evaluation of the effectiveness of AR recommender systems.

[1]http://code.google.com/p/locrec/downloads/list

6. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] M. Allamanis, S. Scellato, and C. Mascolo. Evolution of a location-based online social network: analysis and models. In *Proceedings of the 2012 ACM Internet Measurement Conference*, pages 145–158. ACM, 2012.

[4] R. T. Azuma et al. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[5] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD*, 2007.

[6] S. Benford and L. Fahlén. A spatial model of interaction in large virtual environments. In *Proceedings of the 3rd Conference on European Conference on Computer-Supported Cooperative Work*, pages 109–124. Kluwer Academic Publishers, 1993.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998.

[8] F. Ricci. Mobile recommender systems. *Information Technology & Tourism*, 12(3):205–231, 2010.

[9] F. Ricci and Q. N. Nguyen. Acquiring and revising preferences in a critique-based mobile recommender system. *Intelligent Systems*, 22(3):22–29, 2007.

[10] S. Shang, S. R. Kulkarni, P. W. Cuff, and P. Hui. A random walk based model incorporating social information for recommendations. In *2012 International Workshop on Machine Learning and Signal Processing*, pages 1–6. IEEE, 2012.

[11] K. H. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM Symposium on Applied computing*, pages 1995–1999. ACM, 2008.

[12] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD*, pages 723–732. ACM, 2010.