Real-time Collaborative Filtering Recommender Systems

Huizhi Liang[1,2] **Haoran Du**[2] **Qing Wang**[2]

[1] Department of Computing and Information Systems,
The University of Melbourne,
Victoria 3010, Australia
Email: huizhi.liang@unimelb.edu.au

[2] Research School of Computer Science,
The Australian National University,
Canberra ACT 0200, Australia
Email: duhaoranshux@hotmail.com, Qing.wang@anu.edu.au

Abstract

Recommender systems can help users deal with the information overload issue. Many real-world communities such as social media websites require real-time recommendation making to capture the recent updates of the communities. This brings challenges to existing approaches which mainly build recommendation models at offline. In this paper, we discuss real-time collaborative filtering recommendation approaches. The proposed approaches use locality sensitive hashing (LSH) to construct user or item blocks, which facilitate real-time neighborhood formation and recommendation making. The experiments conducted on a Twitter dataset demonstrate the effectiveness of the proposed approaches.

Keywords: Real-time, Locality Sensitive Hashing, Collaborative Filtering, Recommender System

1 Introduction

Recommender systems is one of the popular personalization applications, which help to solve the information overload issue of users in online communities, i.e., making suggestions regarding which information is most relevant to an individual user. Collaborative filtering approaches such as user-based and item-based K-nearest neighbor methods are widely used to make recommendations in various areas (Adomavicius & Tuzhilin 2005). Collaborative filtering recommender systems usually consist of two phases: (1)An offline model-building phase to build a model storing correlations between users and items. (2) An on-demand recommendation phase that uses the model to make recommendations (Chandramouli et al. 2011).

However, the traditional offline collaborative filtering recommender systems fail to capture the rapid changes of online communities to make real-time recommendations. For example, with the rapid growth of users in social media communities, there are a large number of micro-blog topics emerging every day. They include not only a small number of hot or stream topics but also a large number of less popular topics. Thus, it is important to recommend personally interesting topics to users (Liang et al. 2012). However, since the topics of micro-blogs are constantly changing, it brings difficulty for an offline-built model to capture the latest updates in social media communiteis (Liang et al. 2012).

Neighborhood formation is the key component of collaborative filtering recommender systems. Typically, pair-wise comparisons such as Cosine similarity calculation are commonly used to build the correlations (i.e. find the the nearest neighbors of each user or item). To meet the requirement of real-time response, we need to decrease the number of pair-wise comparisons and find the nearest neighbor users and candidate items quickly. Blocking or indexing techniques can help to significantly decrease the number of comparisons (Christen 2012). The objects in a database can be inserted into one or more blocks according to some blocking criteria, such that only objects within a block are compared with each other. The current blocking techniques are mainly focusing on content features, such as inverted indexing of keywords, and phonetic encoding functions (e.g., Soundex, Double Metaphone) (Christen 2012). Locality sensitive hashing (LSH) (Gionis et al. 1999) is an approximate blocking approach that uses a set of hash functions to map data objects such as users or items within a certain distance range into the same block with a given probability. It can filter out those data objects with low similarities for a given data object, thus decreasing the number of comparisons (Gan et al. 2012). LSH can generate blocks quickly and has advantages such as dimension reduction, noise-tolerant, and similarity-preserving. It has been widely used in industries, such as personalized news recommendation in Google (Li et al. 2011).

We discuss real-time collaborative filtering recommendation approaches. The proposed approaches employ the LSH techniques to construct user and item blocks. Then, we propose approaches to form neighborhood and make recommendations in real-time, based on the generated user and item blocks.

2 Related Work

Recommender systems have been an active research area for more than a decade. The recommendation approaches based on explicit ratings are the major focus. The tasks of recommender systems include rating prediction and top N recommendation. The former

This research was partially funded by the Australian Research Council (ARC), Veda Advantage, and Funnelback Pty. Ltd., under Linkage Project LP100200079. Note the first two authors contributed equally.

task that is to predict the rating value a user will give to a rated item while the latter one is to recommend a set of unrated or new items to the target user (Adomavicius & Tuzhilin 2005). The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are widely used to measure the accuracy of the rating prediction while precision and recall are commonly used for the top N recommendation. For explicit ratings, both tasks are applicable while for implicit ratings, the top N recommendation is more applicable (Adomavicius & Tuzhilin 2005). Recommender systems can be broadly classified into three categories: content-based, collaborative filtering (CF), and hybrid approaches (Adomavicius & Tuzhilin 2005). The user-based and item-based K-nearest neighborhood collaborative filtering are widely used in various application areas.

Approximate blocking techniques such as LSH and tree-based indexing (Bawa et al. 2005) are widely used in nearest neighbor and similarity search in applications such as image search (Dong et al. n.d.), recommender systems (Li et al. 2011), and entity resolution (Kim & Lee n.d., Liang et al. 2014, Li et al. 2013). Recently, some work has been proposed to make real-time recommendations (Chandramouli et al. 2011). For example, Abbar et al. (Abbar et al. 2013) proposed a real-time recommender system for diverse related articles. Li et al. (Li et al. 2012) proposed interest-based real-time content recommendation in online social communities. Diaz-Aviles et al. (Diaz-Aviles et al. 2012) proposed real-time top N matrix factorization recommendation in social streams. Moreover, approximate blocking techniques such as LSH and tree based indexing (Bawa et al. 2005) are used to make efficient news recommendations (Li et al. 2011). However, how to make real-time collaborative filtering recommendations still needs to be explored.

3 Problem Definition

We define some key concepts used in this paper.

- **Users:** $U = \{u_1, u_2, ..., u_{|U|}\}$ contains all users in an online community who have rated or published items.
- **Items (i.e., Products, Topics):** $C = \{c_1, c_2, ..., c_{|C|}\}$ contains all items rated or published by users in U. Items could be any type of online information resources or products in an online community such as web pages, video clips, music tracks, photos, movies, books, topics of micro-blogs (Liang et al. 2012) etc.
- **User profile:** A user profile is a collection of information about a user, such as demographic information, interests or preferences, opinions, friends or other network information. Users' interests or preferences are typical information to profile users. We use binary or numeric weight values for items to represent a user's interests or preferences for items.

Let $u_i \in U$ be a target user, C_{u_i} be the item set that user u_i already has, $\mathbf{C}_{u_i}$ be the candidate item set that are unknown to user u_i, i.e., $\mathbf{C}_{u_i} = C - C_{u_i}$. Let $c_x \in \mathbf{C}_{u_i}$ be a candidate item, $\mathcal{A}(u_i, c_x)$ be the predicted score of how much the user u_i would be interested in item c_x. The problem of top N item recommendation is defined as generating a set of ordered items $c_l, ..., c_m \in \mathbf{C}_{u_i}$ to the use u_i, where $\mathcal{A}(u_i, c_l) \geq ... \geq \mathcal{A}(u_i, c_m)$.

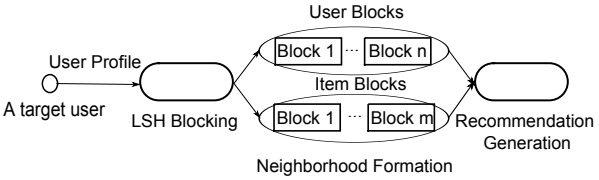


Figure 1: The Framework of Real-time Collaborative Filtering Recommender Systems

4 The Proposed Approach

In this section, we discuss how to conduct real-time collaborative filtering recommendations. The system framework is shown in Figure 1. It describes the key components of real-time collaborative filtering recommender systems, including LSH blocking, neighborhood formation, and recommendation generation.

For real-time user-based collaborative filtering approach, we construct user blocks based on LSH blocking scheme. For a given target user $u_i \in U$, we firstly get the hash signatures of this user based on a LSH family. Then, $u_i \in U$ is allocated to a set of blocks that use the hash signatures as block identifiers. The users that are in the same blocks with $u_i \in U$ are selected as being the neighbor users of u_i. Then, we select the candidate items from the neighbor users and generate a list of recommended items to u_i.

For real-time item-based collaborative filtering approach, we construct item blocks based on a LSH blocking scheme. For a given target user $u_i \in U$, we get the hash signature of each item $c_j \in C_{u_i}$ based on a LSH family. Then, each item $c_j \in C_{u_i}$ is allocated to a set of blocks. The items that are in the same blocks with $c_j \in C_{u_i}$ are selected as being the neighbor items of item c_j. Then, the neighbor items are selected as the candidate items for user $u_i \in U$. The top N ranked candidate items are selected as recommended items for $u_i \in U$.

In the following, we first discuss the LSH blocking scheme that is used to construct user or item blocks based on their Cosine similarities. Then we discuss how to select nearest neighbor users or items based on the generated user or item blocks. After that, we discuss how to make real-time user-based and item-based recommendations.

4.1 LSH Blocking Scheme

Let h denote a hash function for a given distance measure D, $Pr(i)$ denote the probability of an event i, p_1 and p_2 are two probability values, $p_1 > p_2$, $0 \leq p_1, p_2 \leq 1$. h is called (d_1, d_2, p_1, p_2)-sensitive for D, for any data objects x and y, the following conditions hold:

1. if $D(x, y) \leq d_1$ then $Pr(h(x) = h(y)) \geq p_1$
2. if $D(x, y) > d_2$ then $Pr(h(x) = h(y)) \leq p_2$

Popularly used LSH families include the minHash family for Jaccard distance (Anand & Ullman 2011), the random hyperplane projection family for Cosine distance (Anand & Ullman 2011), and the p-stable distribution family for Euclidean Distance (Anand & Ullman 2011). As Cosine distance/similarity is popularly used to measure the similarity of two users or items that are represented as vectors. We discuss how to generate user and item blocks based on a random hyperplane projection family that approximates the Cosine distance/similarity of two vectors.

4.1.1 Random Hyperplane Projection

The random projection method of LSH (Anand & Ullman 2011) is designed to approximate Cosine distance/similarity of any two vectors. The basic idea of this technique is to choose a d-dimensional random hyperplane and use the hyperplane to hash input vectors.

Given an input vector with $n-$dimensions $\vec{x}$, a family $\mathcal{H}_r$ of hash functions such that, for a randomly chosen vector $\overrightarrow{v} \in V$ in a n-dimensional space, a hashing function $h \in \mathcal{H}_r$ is defined as:

$$h(\vec{x}) = \begin{cases} 1 & \text{if } \vec{v} \cdot \vec{x} > 0; \\ 0 & \text{if } \vec{v} \cdot \vec{x} < 0 \end{cases}$$

Each possible choice of $\overrightarrow{v}$ defines a single hash function. This hash function produces a single bit signature for the input vector $\vec{x}$. $\mathcal{H}_r$ contains a set of such functions (i.e., d-dimension) and produces a set of bit signatures. Accordingly, the probability that such hash function family separates two vectors $\vec{x}$ and $\vec{y}$ is directly proportional to the angle between the two vectors (Anand & Ullman 2011):

$$Pr[h(\vec{x}) = h(\vec{y})] = 1 - \frac{\theta(\vec{x}, \vec{y})}{\pi} \quad (1)$$

Following Equation 1, we have,

$$cos(\theta(\vec{x}, \vec{y})) = cos((1 - Pr[h(\vec{x}) = h(\vec{y})])\pi) \quad (2)$$

Thus, Equation 2 provides us a way of approximately calculating Cosine distance/similarity between two vectors. A vector with dimension n is mapped to a binary signature vector with dimension d based on the hash family $\mathcal{H}_r$, usually $d \ll n$. From a probabilistic viewpoint, the more random vectors we use, the more accurate the Cosine distance/similarity between two vectors is. After we get the binary signature of two input vectors $\vec{x}$ and $\vec{y}$, we can use the Hamming distance of their signatures to measure their similarity or distance.

4.1.2 Random Bit Sampling for Hamming Distance

Usually computing the Hamming distance of data objects requires pair-wise similarity/distance calculation. For example, a target user needs to compare with all the other users. To reduce the number of pair-wise comparisons, we can generate blocks for data objects based on their Hamming distance of signatures. We use random bit sampling (Anand & Ullman 2011) to approximate the Hamming distance over d-dimensional signature vectors $\{0,1\}^d$.

A LSH family $\mathcal{F}$ for Hamming distance is simply the family of all the projections of data objects on one of the d coordinates, i.e., $\mathcal{F} = \{h : \{0,1\}^d \rightarrow \{0,1\} \mid h(x) = x_i, i = 1, ..., d\}$, where x_i is the ith coordinate of x. A random function h from $\mathcal{F}$ simply selects a random bit from the input vector. The basic signature is called a length-1 signature and the hash function is called a length-1 hash function.

To amplify the collision probability, given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$, we can construct new families $\mathcal{H}_s$ by the combination of AND-construction or OR-construction of $\mathcal{F}$ (Anand & Ullman 2011). Let $\mathcal{H}_s = \{H_1, H_2, ..., H_l\}$ denote a LSH family that has l number of length-k hash functions. Each length-k hash function is formed by AND-construction of k length-1 hash functions. The l number of hash functions of $\mathcal{H}_s$ has an "OR" relationship between each other. The collision probability of $\mathcal{H}_s$ can be estimated with $p_{k,l} = 1 - (1 - p^k)^l$ where p denotes the collision probability of a length-1 hash function.

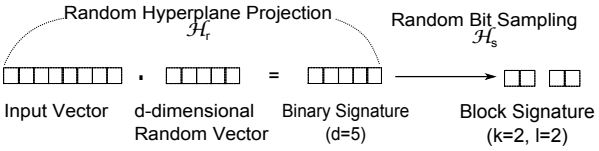


Figure 2: The LSH Blocking Scheme

The LSH blocking scheme $\mathbf{H}$ that are used in this paper consists of two hash families, $\mathbf{H} = \{\mathcal{H}_r, \mathcal{H}_s\}$. Firstly, we use a random hyperplane projection hash family $\mathcal{H}_r$ with d hash functions to get a d-dimensional binary hash signature for each input vector. Then, for each generated hash signature vector, we use a random sampling hash family $\mathcal{H}_s$ to get l length-k hash signatures. $\mathbf{H}$ has three parameters: d, k, and l. Figure 2 illustrate the process of getting the block signature for one input vector.

4.2 Neighborhood Formation

Neighborhood formation is to generate a set of like-minded peers (i.e., similar users) for a target user $u_i \in U$ or a set of similar items for an item $c_i \in C$. This paper adopts the "K-Nearest-Neighbors" technique to find the neighborhood for a user or an item. Typically, the user based K-Nearest-Neighborhood formation approach selects the top K neighbor users with shortest distances to a user u_i through computing the distances between user u_i and all other users of U. While the item-based K-Nearest-Neighborhood formation approach selects the top K neighbor items with the shortest distances to an item c_i through calculating the distances between item c_i and all other items. The distance or similarity measure can be calculated through various kinds of proximity computing approaches such as Cosine similarity and Pearson correlation (Adomavicius & Tuzhilin 2005).

To find the neighborhood of each target user $u_i \in U$ quickly, we construct user blocks and item blocks for users and items respectively. A $|C|$-sized item vector with weight values (denoted as $\vec{u_i}$) which represents user u_i's item preferences is used to profile user u_i. With the LSH blocking scheme $\mathbf{H}$, we can get l hash signatures (denoted as S_i) for user $u_i \in U$. The users that have the same signature will be allocated into the same block. Parameter k decides the similarity threshold (i.e., Hamming distance threshold) of a block. The users in the same blocks with user u_i are the neighbor users of u_i. Thus, we can form neighbourhood quickly via hashing. This approach can filter out users that have low similarities with the target user u_i, thus decreasing the number of pair-wise comparisons. With hashing, we also can update the neighborhood of users quickly after users update their item preferences.

Similarly, we can construct item blocks for each item. For each item c_j, a $|U|$-sized user vector with weight values (denoted as $\vec{c_j}$) is used to represent item c_j. With the LSH blocking scheme $\mathbf{H}$, we can get l hash signatures (denoted as S_j) for item $c_j \in C$. The setting of parameters of $\mathbf{H}$ can be different from the setting to construct user blocks for the purpose of preserving items with different similarity ranges in blocks. The items in the same blocks with item c_j are the neighbor items of c_j.

4.3 Real-time Recommendation Generation

For a given target user $u_i \in U$, with the LSH blocking scheme $\mathbf{H}$, we can generate signatures and construct blocks for each user and each item to form neighborhood. We discuss user-based and item-based real-time recommendation approach.

4.3.1 User-based Recommendation

With users blocks, we can find neighbor users quickly. For a given target user $u_i \in U$, the users in the same user blocks with $u_i \in U$ are the neighbor users of user $u_i \in U$, denoted as $\mathcal{N}_{u_i}$. Typically, we can calculate the pair-wise Cosine similarity of the neighbor user and $u_i \in U$ to select K-nearest neighbor users. However, for large-scale datasets, the number of neighbor users can be large and it is time-consuming to conduct pair-wise comparisons for all neighbor users in the same block with $u_i \in U$. To further decrease the number of candidate neighbors and select a smaller set of nearest neighbors, we count the collision number of each user in all l blocks with user $u_i \in U$ to rank the neighbor users of $u_i \in U$. This is because the number of co-occurrences of a user u_x that appears together with u_i in the blocks reflects the similarity of the two users (Gan et al. 2012). The higher the number of co-occurrences is, the more similar the two users are. We set a threshold φ to select those users that appear at least φ times with the target user u_i together in blocks. Let g_{ix} denote the co-occurrence of user u_x and target user u_i. Let $\mathbf{N}_{u_i}$ denote the selected nearest neighbor record set of u_i. For each user $u_x \in \mathcal{N}_{u_i}$, we add u_x to $\mathbf{N}_{u_i}$ if $g_{ix} > \varphi$, $0 \leq \varphi \leq l$.

Thus, for each target user u_i, we can select top $|\mathbf{N}_{u_i}|$ nearest neighbor users and use a more sophisticated similarity measure approach (i.e., Cosine) to conduct pair-wise similarity calculation. As the time complexity of counting collision number is less than that of other similarity measure approach, the query time can be improved when we employ dynamic collision counting to rank candidate users.

For each target user u_i, a set of candidate items can be generated from the items of user u_i's neighbour users. Let $\mathbf{C}_{u_i}$ denote the the candidate items of user u_i. $\mathbf{C}_{u_i} = \{c_k | c_k \in C_{u_j}, c_k \notin C_{u_i}, u_j \in \mathbf{N}_{u_i}\}$, where C_{u_j} is the items of u_j. Let U_{c_x} denote the user set of item c_x, for each candidate item $c_x \in \mathbf{C}_{u_i}, \mathbf{N}_{u_i} \bigcap U_{c_x}$ is the subset of users in $\mathbf{N}_{u_i}$ who have used item c_x, the prediction score of how much u_i will be interested in $c_x \in \mathbf{C}_{u_i}$ is calculated by considering the similarities between user u_i and those users who are the neighbors of user u_i and have item c_x:

$$\mathcal{A}_u(u_i, c_x) = \sum_{u_j \in \mathbf{N}_{u_i} \bigcap U_{c_x}} \frac{1}{\sqrt{|\mathbf{N}_{u_i} \bigcap U_{c_x}|}} \cdot cosine(\vec{u_i}, \vec{u_j}) \tag{3}$$

The top N items with high prediction scores will be recommended to the target user u_i.

4.3.2 Item-based Recommendation

Similarly, we can generate the top K nearest neighbor items of each item c_j with item blocks. For a given target user $u_i \in U$, let C_{u_i} denote the item set of u_i, the similar items of each item $c_j \in C_{u_i}$ can be used as candidate items for user $u_i \in U$. The candidate item set of item $c_j \in C_{u_i}$ is denoted as $\mathcal{N}_{c_j}$. To further decrease the number of candidate items and select a smaller set of nearest neighbors for each item $c_j \in C_{u_i}$, we count the collision number of each item c_x in all l blocks with item $c_j \in C_{u_i}$ to rank the neighbor items of c_j. Let g_{jx} denote the co-occurrence of item c_x and the item $c_j \in C_{u_i}$ of user u_i. Let $\mathbf{N}_{c_j}$ denote the selected nearest neighbor item set of c_j. For each item $c_x \in \mathcal{N}_{c_j}$, we add c_x to $\mathbf{N}_{c_j}$ if $g_{jx} > \kappa$, $0 \leq \kappa \leq l$. Let $\mathfrak{C}_{u_i}$ denote the selected candidate item set of user u_i, $\mathfrak{C}_{u_i} = \{c_x | c_x \in \mathbf{N}_{c_j}, c_x \notin C_{u_i}, c_j \in C_{u_i}\}$

The prediction score of how much u_i will be interested in item $c_x \in \mathfrak{C}_{u_i}$ is calculated by considering the similarities between item c_x and each item c_j of user u_i:

$$\mathcal{A}_c(u_i, c_x) = \sum_{c_j \in C_{u_i}} \frac{1}{\sqrt{|C_{u_i}|}} \cdot cosine(\vec{c_j}, \vec{c_x}) \tag{4}$$

The top N items with high prediction scores will be recommended to the target user u_i.

5 Experimental Design

In the experiments, we recommend topics to users in a social media community. The experiments were conducted on a real-world dataset crawled from Twitter.com (Liang et al. 2012). After removing the stop words, we select the keywords that are at least used by 5 users as topics. To avoid sparseness, we only selected those users who have used at least 5 topics. The dataset has 2320 users and 3319 topics extracted from 1,214,604 tweets. The user set was split into training and test user sets. We randomly select 10% of users as test user set while the rest 90% users are used as training users. The training user set has 2088 users, the test user set has 232 users. We randomly select 50 items of each test user as the test/answer topic set of this user.

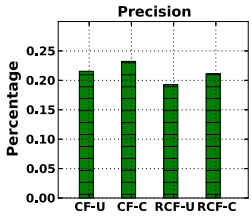
For each test user, the recommender system will generate a list of ordered topics that the test user has not used in his/her training set. The top N topics with high prediction scores will be recommended to the test user. If a topic in the recommendation list was in the test user's test/answer topic set, then this recommended topic was counted as a hit. We adopt Precision and Recall metrics to evaluate the accuracy of recommendations of the proposed approaches. Moreover, to evaluate the efficiency, the average online recommendation time over all test users is used.

6 Experimental Results

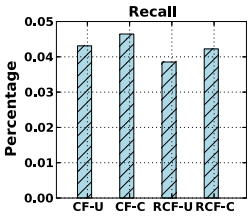
To evaluate the effectiveness of the proposed approaches, we have conducted comparison experiments of the following methods.

- ***CF-U***: The traditional user-based K-Nearest Neighborhood collaborative filtering approach.
- ***RCF-U***: The proposed real-time user-based collaborative filtering approach.
- ***CF-C***: The traditional item-based K-Nearest Neighborhood collaborative filtering approach.
- ***RCF-C***: The proposed real-time item-based collaborative filtering approach.

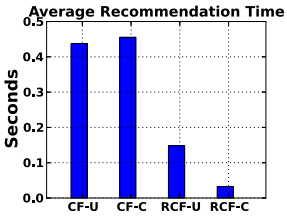
We set $K = 100$ and $N = 10$. For the proposed RCF-U and RCF-C, we set $d = 10$, $k = 4$, and $l = 8$ for hash family $\mathbf{H}$. $\varphi = 2$, $\kappa = 2$. The performance of the compared approaches are shown in Figure 3. The Precision and Recall values of all the approaches are



(a) Precision



(b) Recall



(c) Average Recommendation Time

Figure 3: The comparison results

low. This is because the dataset is sparse. The proposed real-time user-based and item-based collaborative filtering approach RCF-U and RCF-C achieved very close precision and recall results with the traditional collaborative filtering approaches CF-U and CF-C. RCF-U and RCF-C conducted much quicker recommendations. This can be explained that the proposed approaches largely decreased the number of pair-wise comparisons. They can incrementally and efficiently identify nearest neighbors when new updates occur in the community. Thus, they can be used for real-time recommendation making in online communities which requires quick responses for users' updates such as social media communities.

7 Conclusions

We discussed a real-time user-based and item-based collaborative filtering recommendation approach. To facilitate real-time recommendation, we adopt a LSH family that approximates Cosine distance/similarity and a LSH family that approximates Hamming distance to construct user and item blocks. Then, we discussed how to identify a set of nearest neighbors efficiently and how to rank candidate items quickly. As LSH techniques can be used for various types of item contents (e.g., text, image, numeric or binary weight values), this approach is applicable for various kinds of communities, especially those communities that have items with high dimensional content information and require quick recommendation responses to users updates. The experiments were conducted on a Twitter dataset and make real-time topic recommendations. The future work will consider the temporal effect of items and users.

References

Abbar, S., Amer-Yahia, S., Indyk, P. & Mahabadi, S. (2013), Real-time recommendation of diverse related articles, *in* 'WWW', pp. 1–12.

Adomavicius, G. & Tuzhilin, A. (2005), 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *TKDE* **17**(6), 734–749.

Anand, R. & Ullman, J. D. (2011), *Mining of massive datasets*, Cambridge University Press.

Bawa, M., Condie, T. & Ganesan, P. (2005), LSH forest: self-tuning indexes for similarity search, *in* 'WWW', pp. 651–660.

Chandramouli, B., Levandoski, J. J., Eldawy, A. & Mokbel, M. F. (2011), Streamrec: A real-time recommender system, *in* 'SIGMOD', pp. 1243–1246.

Christen, P. (2012), *Data Matching*, Data-Centric Systems and Appl., Springer.

Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L. & Nejdl, W. (2012), Real-time top-n recommendation in social streams, *in* 'RecSys', pp. 59–66.

Dong, W., Wang, Z., Josephson, W., Charikar, M. & Li, K. (n.d.), Modeling lsh for performance tuning, *in* 'CIKM'08', ACM, pp. 669–678.

Gan, J., Feng, J., Fang, Q. & Ng, W. (2012), Locality-sensitive hashing scheme based on dynamic collision counting, *in* 'SIGMOD', pp. 541–552.

Gionis, A., Indyk, P., Motwani, R. et al. (1999), Similarity search in high dimensions via hashing, *in* 'VLDB', pp. 518–529.

Kim, H.-S. & Lee, D. (n.d.), HARRA: fast iterative hashed record linkage for large-scale data collections, *in* 'EDBT'10', ACM, pp. 525–536.

Li, D., Lv, Q., Xie, X., Shang, L., Xia, H., Lu, T. & Gu, N. (2012), 'Interest-based real-time content recommendation in online social communities', *Knowledge-Based System* **28**, 1–12.

Li, L., Wang, D., Li, T., Knox, D. & Padmanabhan, B. (2011), Scene: a scalable two-stage personalized news recommendation system., *in* 'SIGIR', pp. 125–134.

Li, S., Liang, H. & Ramadan, B. (2013), Two stage similarity-aware indexing for large-scale real-time entity resolution, *in* 'AusDM 2013', CRPIT, Vol. 146.

Liang, H., Wang, Y., Christen, P. & Gayler, R. W. (2014), Noise-tolerant approximate blocking for dynamic real-time entity resolution, *in* 'PAKDD', pp. 449–460.

Liang, H., Xu, Y., Tjondronegoro, D. & Christen, P. (2012), Time-aware topic recommendation based on micro-blogs, *in* 'CIKM', pp. 1657–1661.