Interest Prediction on Multinomial, Time-Evolving Social Graphs

Nozomi Nori, Danushka Bollegala and **Mitsuru Ishizuka**
Graduate School of Information Science & Technology, The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan.

Abstract

We propose a method to predict users' interests in social media, using time-evolving, multinomial relational data. We exploit various actions performed by users, and their preferences to predict user interests. Actions performed by users in social media such as Twitter, Delicious and Facebook have two fundamental properties. (a) User actions can be represented as high-dimensional or multinomial relations - e.g. referring URLs, bookmarking and tagging, clicking a favorite button on a post etc. (b) User actions are time-varying and user-specific – each user has unique preferences that change over time. Consequently, it is appropriate to represent each user's action at some point in time as a multinomial relational data. We propose *ActionGraph*, a novel graph representation for modeling users' multinomial, time-varying actions. Each user's action at some time point is represented by an *action node*. ActionGraph is a bipartite graph whose edges connect an action node to its involving entities, referred to as object nodes. Using real-world social media data, we empirically justify the proposed graph structure. Our experimental results show that the proposed ActionGraph improves the accuracy in a user interest prediction task by outperforming several baselines including standard tensor analysis, a previously proposed state-of-the-art LDA-based method and other graph-based variants. Moreover, the proposed method shows robust performances in the presence of sparse data.

1 Introduction

The World Wide Web contains numerous relations among users and *resources* referred to by URLs. The number of relational data has increased rapidly over the past few years as a result of the rapid growth of social media such as social networking services. In social media, users perform various actions such as expressing their interests in a particular website in Facebook[1], *retweeting* a comment made by a friend in Twitter[2]. The manners in which different users express their interests on a particular resource vary across different social media. For example, a user might bookmark a website in Delicious[3] when she wants to share it with others, whereas another user might *tweet* it in Twitter. The different actions performed by users in different social media provide valuable clues for predicting user interest in numerous Web-related tasks such as information recommendation. For example, if two users A and B perform similar actions towards numerous resources, then it is likely that A and B have similar interests. Consequently, a recommender system can use this information to recommend information (e.g. news, products, movies etc.) to B based on the interests expressed by A. Moreover, by exploiting the interest related actions we can overcome the *cold-start* problem [Jamali and Ester, 2009] which we frequently encounter in recommendation systems. So-called cold-start users, that have rated only a very small number of items, might have some data describing their interests in some social media that can be utilized in a recommendation system.

Despite the importance of user interest prediction in social media, it remains a relatively under-studied problem that poses two main challenges. First, the actions performed by users in social media involve high-dimensional and multinomial relations. For example, a user might express her interest on a particular news article by *tweeting* it in Twitter, which then gets marked as a *favorite* by a different user. Such an action involves multiple entities including multiple users and multiple resources, which calls for a multinomial representation. The number of users as well as the number of resources on the Web is extremely large, which potentially results in a large number of relation instances being created over time. To accurately capture user interest from this high-dimensional relational data, we must develop a method that is robust to data sparseness.

Second, the interest of users in the Web is a dynamic phenomenon that changes over time. For such time-aware recommendations, it has been pointed out that capturing users' temporal preferences is important [Xiang *et al.*, 2010]. Overall behaviors of a user can be characterized by her long-term preferences. But at any given time, a user is also influenced by her short-term interests that only last for few days such

[1] www.facebook.com

[2] www.twitter.com

[3] www.delicious.com

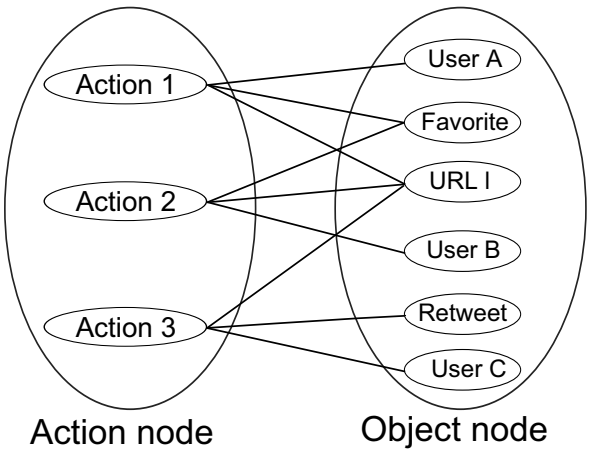

Figure 1: An example of ActionGraph representation.

as traveling or birthdays. To capture users' temporal preferences, it is pointed out that capturing *user-specific* time scale is more effective [Xiang *et al.*, 2010]. Although many time-evolving models introduce time as a universal dimension shared by all users, in some cases we can observe local effects that involve only a specific user. Although this user-specific time scale is an important aspect of time-varying systems, it has not been studied adequately. We propose a method that captures both multinomial and time-dependent, user-specific actions in social media. Our contributions in this paper are summarized as follows.

- We propose *ActionGraph*, a novel graph representation for modeling multinomial, time-dependent actions performed by users in social media.
- Using real-world social media data, we empirically justify the proposed graph structure. We show that adding features such as keywords based on other action types can be effective for interest predictions. This will support ActionGraph structure, which can model multinomial relations.
- To evaluate the usefulness of the proposed ActionGraph, we employ it in a user interest prediction task in social media. Our experimental results show that the ActionGraph-based user interest prediction method outperforms several baseline methods in terms of precision and robustness against data sparseness.

2 Proposed Method

2.1 ActionGraph Construction

An action performed by a user in a social media often involves multiple entities. For example, let us consider the scenario where a user A tags a website referred to by URL l with the keyword, "artificial intelligence". We can view this action as involving four entities: user A, URL l, the action verb, or predicate "tag", and the keyword "artificial intelligence". To represent this action, we create an *action node*, which corresponds to the action itself, and create four *object nodes*, which correspond to the involved entities, user A, URL l, the action verb, or predicate "tag", and the keyword "artificial intelligence". Next, we connect the action node to each of those object nodes. ActionGraph is defined as a graph $G = (V_{AC} \cup V_{OB}, E)$, where each vertex in V_{AC} corresponds to a specific action at some point in time represented by an *action node*, and each vertex in V_{OB} corresponds to an entity involved in an action represented by an *object node*. An edge in $e \in E$ connects a vertex in V_{AC} to a vertex in V_{OB}. ActionGraph is a bipartite graph whose edge connects an action node and the object nodes involved in a particular action. Figure 2 shows an example of users' actions in social media. We refer to the type of actions a user takes to a specific object as *action type*. Figure 1 shows the corresponding ActionGraph.

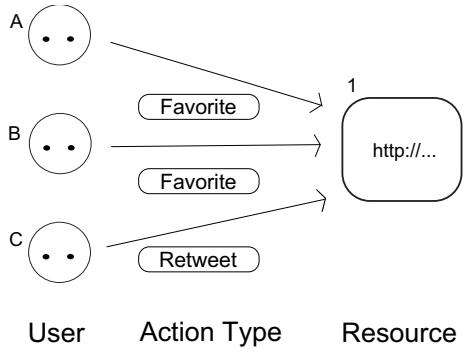

Figure 2: An example of users' actions in social media.

ActionGraph representation preserves the original co-occurrences among multinomial data, because by following the links from an action node, one can retrieve all the entities that are involved in the action represented by the action node. Furthermore, ActionGraph preserves the time information for each user, because an action node corresponds to a specific point in time in which that action is performed by the user. For example, if a user performed one action now and another action after several days, those two actions will be represented by *different* action nodes.

2.2 Predicting User Interest using an ActionGraph

In recommendation tasks, *collaborative filtering* [Resnick *et al.*, 1994; Shardanand and Maes, 1995] is a major approach nowadays. Collaborative filtering models the recommendation task as computing the similarities between a user and a set of items. For example, two users are considered to be similar if the set of items those users are involved in are similar. Then one can be recommended items that similar users preferred but s/he has not yet involved in. This basic idea can be applied to our case, ActionGraph. It is natural to consider that two users are similar if their actions are similar. So following these previous works, we model the problem of predicting the user interest in a resource as a problem of computing the similarity between a user and a set of resources, by exploiting their actions. Given an ActionGraph, to predict potentially interesting resources to a particular user, we propose a similarity function that exploits the structure of the ActionGraph using *graph kernels* [Fouss *et al.*, 2006]. Because object nodes include users and resources, we must measure similarities between object nodes. Consequently, we define

similarities between action nodes in terms of object nodes, and define similarities between object nodes in terms of action nodes as described below.

Similarities between action nodes:
Two action nodes are considered to be similar if the sets of involved object nodes are similar.

Similarities between object nodes:
Two object nodes are considered to be similar if action nodes which involve those object nodes are similar.

We use a graph kernel to incorporate the above-mentioned criteria and compute the similarity between a user and a resource in an ActionGraph. Graph kernels enable us to exploit the structural properties in an ActionGraph. Graph kernels have the desirable properties that the similarity between two nodes increases concomitant with number of paths connecting those nodes, and the similarity between two nodes decreases when the *length* of the connecting paths increases. Consequently, two nodes are considered as similar if there are many short paths connecting them. Several graph kernels has been proposed in the literature. Among them, graph kernels based on *Laplacian matrix* such as the *regularized Laplacian kernel* [Smola and Kondor, 2003], are suitable to find similar nodes [Ito *et al.*, 2005]. Laplacian matrix is a matrix representation of a graph. The unnormalized Laplacian matrix of matrix $\mathbf{M}$ is defined as $\mathbf{L}(\mathbf{M}) = \mathbf{D}(\mathbf{M}) - \mathbf{M}$, where $\mathbf{D}(\mathbf{M})$ is a diagonal matrix which has its diagonal elements set to that in a matrix $\mathbf{M}$ and all other elements to zero. The unnormalized Laplacian matrix can be normalized by multiplying $\mathbf{D}(\mathbf{M})^{-1/2}$ from left and right sides to $\mathbf{L}(\mathbf{M})$ to give a symmetric normalized Laplacian matrix: $\mathbf{D}(\mathbf{M})^{-1/2}\mathbf{L}(\mathbf{M})\mathbf{D}(\mathbf{M})^{-1/2}$.

The regularized Laplacian kernel RL_β is defined as follows,

$$\mathbf{RL}_\beta(\mathbf{AA}^\top) = \sum_{k=0}^{\infty}(-\beta\mathbf{L}(\mathbf{AA}^\top))^k = (\mathbf{I} + \beta\mathbf{L}(\mathbf{AA}^\top))^{-1} \quad (1)$$

where $\mathbf{A}$ is a similarity matrix and $\mathbf{I}$ is the identity matrix, and β is a parameter used to tune how much similarity weights are placed on distant nodes when measuring the similarity between a pair of nodes. The similarity matrix $\mathbf{A}$ is constructed from one ActionGraph such as Figure 1. Besides the regularized Laplacian kernel there can be other graph kernels suitable for the task of predicting user interest using an ActionGraph. However, in this paper we limit the discussion to the above-mentioned regularized Laplacian kernel for its simplicity.

3 Experiments

Two types of experiments are conducted to empirically evaluate the proposed method. First, we conduct an analysis to verify whether mixing different action types are effective for interest prediction tasks. Second, we empirically evaluate the proposed ActionGraph by using it in a prediction task.

3.1 Social Media Analysis

We designed our experiments as analyzing relations between different types of actions. Using *favorite* data in Twitter, how precisely can we predict another action such as tagging on Delicious or *retweet* in Twitter? If we can predict well, it will support to aggregate these two kinds of data.

Table 1: Summary of the relational tuples in Twitter and Delicious dataset for action types analyses.

Action type	Facets	Tuples
tweet	(user, URL, tweet id)	213,929
retweet	(user, URL, original tweet id)	22,680
favorite	(user, URL, original tweet id)	37,912
following	(user, user)	214,561
tagging	(user, URL, keywords)	21,530

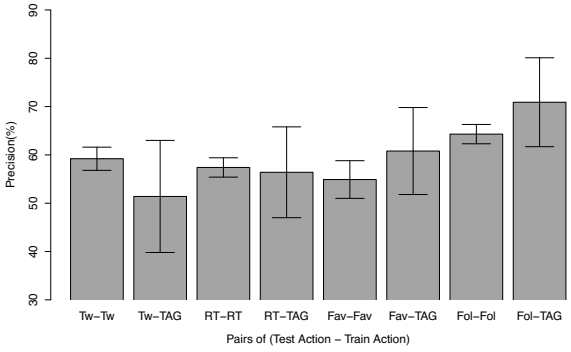

Figure 3: Predicting twitter actions by Delicious tagging.

Dataset

We collected a dataset from Twitter and Delicious. Twitter is one of the most popular social media. When we try to aggregate data from multiple web services, we need some *linking* entities that link different applications. URL is one of such kind of entities. In addition, URL is an entity that can indicate various resources. So it will be suitable to choose action types which involve URLs. We chose four major functions, *tweet*, *retweet*, *favorite* and *following*, which meet the requirement. We show some details about those functions below. Users can *tweet* a short post, called *tweet*. Users can write texts including URLs in their tweets. By *following* another user, users can view that user's tweets. *Retweet* is a function to re-post other user's tweet. Users can also *favorite* a post. We also chose Delicious, a social bookmarking web service. We picked users' *tagging* actions from Delicious. We started crawling from a specific user, and then we followed the users' following networks to two hop links. Then using *FriendFeed*[4] data, we identified users who have also FriendFeed accounts. Some of those users have delicious accounts, which can be identified by FriendFeed. We collected data from these identified users with time stamps ranging from August 1 2010 to August 30 2010. The relational tuples are summarized in Table 1.

Prediction setting

We conducted preliminary experiments, and segmented the duration every three/nine/twenty-seven days. The performance was best with every three days dataset, so we segment

[4] www.friendfeed.com

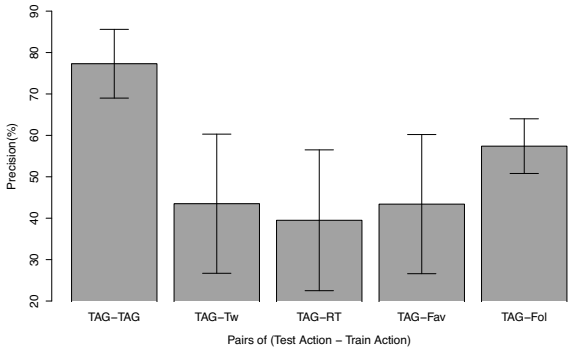

Figure 4: Predicting Delicious tagging by twitter actions.

the duration into 10 time slots (every three days). In the following we shall use $t \in [1, 9]$ to denote a time slot index. We used slot $t \in [1, 10]$ data as train data, and $t+1 \in [2, 10]$ data as test data. The task is a binary classification and we evaluate the precision. Input data is relational tuples in Table 1, each of them corresponding to one action. Features of retweet actions are (user, "retweet", URL, original tweet id). We made 50.0% negatives for each train dataset, so a random prediction achieves 50.0% precision. We adopted *L2 regularized logistic regression* using a machine learning library *Classias*[5] . To compare various type of actions, which vary in the amount of data, we made the amount of each test/train data equal in each time slot.

Results and Discussion

Figure 3 shows a result predicting Twitter actions by Delicious tagging and Figure 4 shows a result predicting Delicious tagging by Twitter actions. We note tweet, retweet, favorite, following, tagging as Tw, RT, Fav, Fol, TAG respectively in Figure 3 and 4. Each bar noted as A-B shows an average precision on each time slot for the experiment that an action type for test data is A and the train action type is B. An error bar shows standard deviation. When we predict favorite/following by using tagging, it seems to achieve almost equal or a little higher performances in a mean viewpoint. This means that tagging can substitutes favorite/following actions, suggesting that mixing different action types can be effective for sparse data. However, predicting Delicious tagging by Twitter actions results in rather lower performances as showed in Figure 4. We think one of the reason of this asymmetry is that keywords reflect users' preferences that complement preferences reflected in actions without keywords. So even if the action types are different, adding features such as keywords based on other action types will be effective for interest predictions. This will support our graph structure because ActionGraph can preserve the co-occurrences among multinomial data.

3.2 Evaluations via Prediction

Dataset

The crawling condition is same as the previous analysis. The summary of datasets is in Table 2. *Original-tweet-user* means the user who first tweeted the post and it gets retweeted or favorited by other users.

[5] www.chokkan.org/software/classias/index.html.en

Table 2: Summary of the relational tuples in Twitter dataset for prediction tasks.

Action type	Facets	Tuples
Retweet	(user, URL, original-tweet-user)	14,392
Favorite	(user, URL, original-tweet-user)	25,884

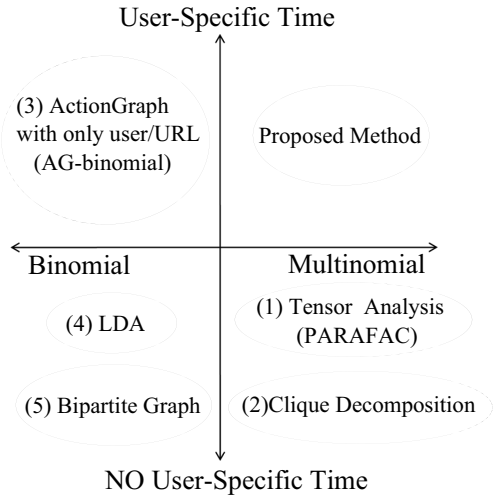

Figure 5: Baseline Methods.

Prediction setting

When a user takes a certain action to a resource, we hypothesized that the user gets interested in the resource. We define the interest prediction problem here as predicting resources rank list which a given user will get interested in. The input is relational tuples, and given tuples, we can create ActionGraph. The output is similarities between entities. We predict how likely a user gets interested in resources based on the similarities between user and resource. In this experiment, we used users' retweet and favorite actions. We segment the duration into 10 time slots following the previous setting.

Baseline methods

We compared our proposed method to five methods. Our proposed method models two properties, multinomial relations and user-specific time scale. So we designed baseline methods in terms of those two properties. Figure 5 shows the overview of baseline methods. (1) Standard tensor analysis (*PARAFAC*) [Bader and Kolda, 2006] and (2) *Clique Decomposition* can deal with multinomial relations but cannot user-specific time scale. (3) ActionGraph with only user/URL (*AG-binomial*) can deal with user-specific time but cannot multinomial relations. (4) latent Dirichlet allocation [Blei *et al.*, 2003] (*LDA*) and (5) *Bipartite Graph* cannot handle either multinomial relations or user-specific time. In the following, we explain those methods in details. *PARAFAC* is one of the most popular tensor decompositions. A tensor is a mathematical representation of a multi-way array, and tensor decomposition is a form of higher-order principal component analysis. With PARAFAC, each dimension is projected

on latent spaces. In this experiment, users and URLs are first projected on the latent spaces, and then the prediction is made based on the cosine similarities of the projected vectors of users and URLs. *Clique Decomposition* decomposes co-occurrences of multiple entities into pair-wise edge on a graph. Similarities between nodes are computed by the same algorithm as our proposed method, the regularized Laplacian kernel. About three graph-based methods, (2), (3) and (5), we apply the regularized Laplacian kernel. The only difference between our proposed method is how we construct a graph. So those three methods are useful to see how the way of constructing a graph affects the performances of prediction tasks. *AG-binomial* constructs ActionGraph but exploits only user/URL features. *LDA* is a generative probabilistic model that introduces latent variables and allows sets of observations to be explained by unobserved, hidden variables which often called as *topics*. The probability a user u gets interested in a resource (URL) r_i can be formalized as $P(r_i|u) = \sum_{j=1}^{Z} P(r_i|z_i = j)P(z_i = j|u)$, where $P(r_i|u)$ is the probability of the i-th URL for a given user u and z_i is the latent topic. *Bipartite Graph* develops a graph whose edge connects only user nodes and URL nodes.

Evaluation metrics

We use two types of evaluation metrics, *R-Precision* for the accuracy evaluation and *Coverage* for the robustness evaluation against data sparseness. *R-Precision* is the precision at rank R, where R is the number of the total *true* data. The number of URLs referring actions varies according to users, so R-Precision will be suitable for our measure. We averaged R-Precision for each user and each slot. *Coverage* is the percentage of $\langle$user, URL$\rangle$ pairs in the test set for which we can compute a recommendation. This information helps to make recommendations for cold start users.

Parameter setting

We used 20% of all data as a preliminary dataset, and with this dataset we optimized each parameter in terms of R-Precision, for both our method and baseline methods. In our prediction method, we changed the parameter β in the equation 1 as $0.01, 0.05, 0.1$. The results stayed about the same, but results where $\beta = 0.1$ and 0.01 were same and had a little better performances, so we fixed β as 0.1. Other graph-based methods followed this parameter setting. It is pointed out that the regularized Laplacian kernel is stable over β [Ito *et al.*, 2005], and actually it was true in our experiment. In PARAFAC, we changed the reduced latent dimension as $200, 400, 600, 800$ and got 600. In LDA, we changed the topic number as $100, 200, 400, 600$, and got 400. There are other two parameters, α and β in LDA. When applying LDA to a generative modeling of documents, α is a parameter of the uniform Dirichlet prior on the per-document topic distributions, and β is a parameter of the uniform Dirichlet prior on the per-topic word distribution. About β, we changed it as $0.01, 0.05, 0.1$, and got 0.01. It is pointed out that changing α by the topic number is effective [Wallach *et al.*, 2010], so we set α for each topic as $\frac{50}{topicNumber}$ following the pLDA[6] implementation we used.

[6] www.code.google.com/p/plda

Results and Discussion

The results are given in Table 3. In R-Precision, ActionGraph outperforms all baselines. The reason can be thought that both multinomial relations and user-specific time scale are actually important to predict user interests. In Coverage, ActionGraph significantly outperforms PARAFAC and two graph-based methods for binomial relations, Bipartite Graph and AG-binomial. Actiongraph can compute almost all similarities for possible pairs. Even users and URLs are sparse, action types are tend to be shared by many action nodes, so action types may serve as a *bridge* among other nodes and relax graph sparseness. So ActionGraph is expected to be robust to data sparseness. It is also noteworthy that ActionGraph outperforms all graph-based methods. Although many papers have been written about capturing similarity of users/resources using graph models, our research shows that an important factor for the performance of the algorithm is the choice of the graph itself.

4 Related Work

4.1 Multinomial Relation Analysis

By multinomial relation analyses, the co-occurrence of more than three entities can be captured. This is useful in many situations. For example, even if two users tagged a same keyword to a same document, the meaning of the tag can be different for each user. If one preserves the co-occurrence of not only keywords and documents, but also of users, it is expected more meaningful mining will be possible. Existing techniques for multinomial analyses include pair-wise analyses and tensor based analyses. The pair-wise kernel [Ben-Hur and Noble, 2005] was proposed for predicting protein-protein interactions. The same kernel was proposed for entity resolution [Oyama and Manning, 2004], and collaborative filtering [Basilico and Hofmann, 2004], independently. [Zhou *et al.*, 2008] deals with three entities, author, paper and venue, and decomposes them into three bipartite graphs. But these pair-wise methods involve the loss of valuable information, the original co-occurrence among more than three entities. Tensor-based approaches have been investigated by several areas, too. [Lin *et al.*, 2009] generalized NMF (Non-negative matrix factorization) [Lee and Seung, 2001] in the case of tensors and proposed a method to discover community evolutions in social media. Tensor-based analyses have a challenge to data sparseness problems. These approaches are promising, but they are not explicitly modeling user-specific time dimensions.

4.2 Time-Evolving Network Analysis

There are works [Sun *et al.*, 2007] that handle time as an universal dimension. But it will be more preferable to deal with time as more local, user-specific ones in some situations. [Xiang *et al.*, 2010] is a new approach to this problem. They introduce *session nodes* that connect item nodes bought by a same user in a session - e.g. in a day or in one week. But this approach cannot handle multi-dimensional data. In social media, data consists of multiple co-evolving dimensions. For instance, a user's action such as uploading photos can trigger another user's different action such as bookmarking and

Table 3: The average prediction performances (averaged on user and time slot) for evaluations via prediction.

	Proposed Method	PARAFAC	Clique Decomposition	AG-binomial	LDA	Bipartite Graph
R-Precision	7.6±3.3 %	3.4±2.1 %	4.2±1.6 %	4.2±2.6 %	4.3±1.4 %	2.0±1.1 %
Coverage	99.8±0.0 %	43.6±6.7 %	99.8±0.0 %	41.9±7.9 %	99.0±0.2 %@	56.4±10.8 %

tagging them. So time-evolving approaches should be able to handle multi-dimensional data.

5 Conclusion

We proposed ActionGraph, a new graph representation for modeling users' multinomial, time-evolving actions. Our method has both higher precision and higher coverage compared to several baseline methods, and expected to be effective for sparse data. As a future work, we plan to exploit user-specific time scale more expressly. For instance, if we bundle action nodes a specific user performed in a session, e.g. - in a day, we will be able to capture users' temporal preferences, which is similar to the previous work [Xiang *et al.*, 2010].

References

[Bader and Kolda, 2006] Brett W. Bader and Tamara G. Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. In *TOMS 32(4)*, pages 635–653, 2006.

[Basilico and Hofmann, 2004] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proc. 21st International Conference on Machine Learning (ICML 2004)*, 2004.

[Ben-Hur and Noble, 2005] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein-protein interactions. In *Proc. 13th International Conference on Intelligent Systems for Molecular Biology. Bioinformatics 21*, pages i38–i46, 2005.

[Blei *et al.*, 2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Fouss *et al.*, 2006] Francois Fouss, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proc. 6th International Conference on Data Mining (ICDM 2006)*, pages 863–868, 2006.

[Ito *et al.*, 2005] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. Application of kernels to link analysis. In *Proc. 11th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2005)*, pages 586–592, 2005.

[Jamali and Ester, 2009] Mohsen Jamali and Martin Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proc. 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009)*, 2009.

[Lee and Seung, 2001] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances Neural Information ProcessingSystems*, 13:556–562, 2001.

[Lin *et al.*, 2009] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proc. 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009)*, 2009.

[Oyama and Manning, 2004] Satoshi Oyama and Christopher D. Manning. Using feature conjunctions across examples for learning pairwise classifiers. In *Proc. 15th European Conference on Machine Learning (ECML 2004)*, pages 322–333, 2004.

[Resnick *et al.*, 1994] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of The Conf. on Computer Supported Cooperative Work*, 1994.

[Shardanand and Maes, 1995] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating gword of mouthh. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 1995.

[Smola and Kondor, 2003] Alex J. Smola and Risi Kondor. *Kernels and regularization on graphs*. Springer, 2003.

[Sun *et al.*, 2007] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proc. 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 687–696, 2007.

[Wallach *et al.*, 2010] Hanna M. Wallach, David Mimno, and Andrew Mccallum. Rethinking lda: Why priors matter. In *In Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS 2009)*, 2010.

[Xiang *et al.*, 2010] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proc. 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2010)*, 2010.

[Zhou *et al.*, 2008] Ding Zhou, Shenghuo Zhu, Kai Yu, Xiaodan Song, Belle L. Tseng, Hongyuan Zha, and C. Lee Giles. Learning multiple graphs for document recommendations. In *Proc. 17th International World Wide Web Conference (WWW 2008)*, pages 141–150, 2008.