

Exploiting User Interest on Social Media for Aggregating Diverse Data and Predicting Interest

Nozomi Nori and 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.
 nozomi.nori@gmail.com, danushka@iba.t.u-tokyo.ac.jp, ishizuka@i.u-tokyo.ac.jp

Abstract

More and more users have been taking various actions to diverse resources referred to by URLs such as news, web pages, images, products, movies as a result of the growth of social media. They are annotating, *tweeting* in Twitter, *reblogging* in Tumblr, and *Liking* in Facebook, etc. Analyses about these diverse actions will be useful for aggregating or integrating diverse resources. In this paper, we view users' actions to resources as expressing their some interests, and by investigating how their interests are expressed in social media, we get suggestions for aggregations. Our results show that a certain kind of action (such as tagging on Delicious) can be used to make predictions on a different kind of action (such as *favorite* on Twitter). These analyses will be useful for aggregating or integrating diverse contents on multiple sources. In addition to some experimental analyses, we propose a novel method to predict users' interests in social media, using time-evolving, multinomial relational data. Our experimental results show that the proposed method significantly outperforms standard tensor analysis and an existing state-of-the-art method (LDA) in prediction tasks.

Introduction

In social media, users perform various actions within their social networks such as expressing their interests to a particular website in Facebook¹ or *retweeting* a comment made by a friend in Twitter². The manners in which different users express their interests to a particular resource vary across social media. For example, a user might bookmark a website on Delicious³ when she wants to keep a record of it, whereas another user might *favotite* a resource in Twitter. When different users take some actions to a certain resource, we can hypothesize that those users get interested in the resource although their action types may be different each other. When a user takes some actions to a resource, we can view that user is interested in the resource, whether it is not clear she did it because she loved it or she wanted to call in the question, or so. We view users' behaviors as Figure 1. Each user has unique preferences which are

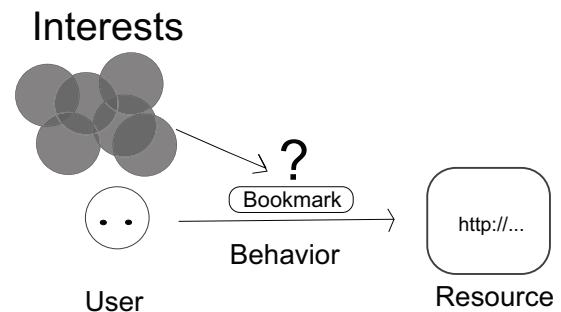


Figure 1: A model of users' behaviors in social media.

influenced by numerous factors such as her cultural background, education, and past experience and so on. Consequently, the different actions performed by users in social media provide valuable clues for extracting users' preferences, predicting users' interests in numerous Web-related tasks such as information recommendation and trust network extraction (Matsuo and Yamamoto 2009). For example, if two users *A* and *B* perform similar actions to numerous resources, then it is likely that *A* and *B* have similar interests. As a result, 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; Massa and Bhattacharjee 2005) in recommendation systems. So-called cold-start users who have rated only a very small number of items, are expected to have some interests data on the Web such as social networks.

Despite the importance of user interest prediction in social media, it is a relatively under-studied problem that poses three main challenges. First, how can we integrate or aggregate various action data existing on the vast Web? There are various web services and functions provided on the Web, and more and more users use various web services and functions. Even when we focus attention on one user, the user is likely to use more than one web service or function. Both the amount and the variety of data will continue to increase. How can users get meaningful information from these data? What kind of suggestions can we get from those data for aggregating diverse data? For example, is it really effective

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

¹www.facebook.com

²www.twitter.com

³www.delicious.com

to mix different types of actions? If so, what kind of actions are useful to predict a certain kind of action? To answer these questions, we develop experimental analyses. We examined whether a certain kind of action such as *favorite* on Twitter can be exploited to predict another kind of actions such as tagging on Delicious.

Second, the actions performed by users in social media involve high-dimensional and multinomial relations (Lin et al. 2009). For example, a user might express her interest in a particular news item by *retweeting* it in Twitter, which then gets marked as *favorited* by a different user. Such an action involves 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 results in a large number of relation instances. To accurately capture user interest from this high-dimensional multinomial relational data, we need a method that is both efficient and robust to data sparseness.

Third, the interest of users on the Web is a dynamic phenomenon that constantly varies over time and from one user to another. For such time-aware recommendations, it has been pointed out that capturing users' temporal preferences is important (Xiang et al. 2010). Overall behavior of a user may be characterized by her long-term preferences. But at any given time, a user is also affected by her short-term interest due to personal events such as travellings or birthdays etc. 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 (Sun et al. 2007; Sun, Tao, and Faloutsos 2006) introduce time as an universal dimension shared by all users, in some cases we can observe local effects that involve only a specific user or some specific users. Although this user specific time scale is an important aspect of time-varying systems, there is much less work about it.

For latter two challenges, we propose a new method that captures both multinomial and time-dependent, user-specific actions in social media.

Our contributions are summarized as follows.

- We conduct experimental analyses about the relations among diverse actions on social media. Our results show a specific action such as tagging on Delicious can be exploited to predict another action such as *favorite* on Twitter. These analyses will be useful for aggregating or integrating diverse contents on multiple services.
- We propose *ActionGraph*, a novel graph representation for modeling multinomial, time-dependent actions performed by users in social media. To the best of our knowledge, this method is the first one that captures both multinomial and user-specific time scale.
- Our experimental results show that the proposed method significantly outperforms tensor analysis and a previously proposed state-of-the-art prediction method based on Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003). The proposed method is not only precise but also robust to data sparseness.

The rest of the paper is organized as follows. Firstly, we show our experimental analyses about the relations among

diverse actions. The results will be useful for integrating or aggregating multiple sources on the Web. Secondly, we present a method for interests predictions that captures both multinomial and user-specific time scale. Thirdly using real-world data, we demonstrate the utility of our method. Then we review some related work and conclude.

Diverse actions analyses

In this section, we investigate how users' interests are expressed in social media, and get suggestions for integrating or aggregating multiple resources. 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. Figure 6 shows an example of users' actions on social media. In the following, we shall call what type of action a user takes to a specific resource as *action type*.

Experiments

Dataset We have collected a dataset from Twitter and Delicious. Crawled Dataset will be published at the author's web site⁴. Twitter is one of the most popular and growing social media, and can be thought as a suitable application for our Social Media analyses. We are aiming to aggregate data from multiple web services, so we need some *linking* or *bridging* entities that link/bridge different applications. URL is one of the entities which link data from one application to another application. Besides URL is an entity that can indicate various resources. So it will be suitable to choose action types which involve URLs. Major functions in Twitter, *tweet*, *retweet*, *favorite* and *following* meet this requirement. So we choose those four functions for Twitter functions. 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, by clicking one button. By retweeting a post, the user's followers can view the post. Users can also *favorite* a post by clicking one button. We chose another application, Delicious, for our analyses. Delicious is a social bookmarking web service. We picked users' tagging actions from Delicious. Delicious tagging actions are similar to URLs referring actions in Twitter, in the sense that they are referring URLs actions. We started crawling from a specific user, then we followed the users' following networks to two hop links. Then using *FriendFeed*⁵ data, we identified users who also have 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. Tagging action on Delicious is noted as tagging.

⁴nozomi.shi-ba.org

⁵www.friendfeed.com

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

Prediction setting We conducted preliminary experiments, and segment the duration every three days, every nine days, every twenty-seven days. The prediction performance was best with every three days dataset. Twitter or Delicious can be thought as a real-time service, so short segments such as every three days may be preferable. So we segment 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 $t[1 - 9]$ data as training data, and $t + 1[2 - 10]$ data as testing data. The task is a binary classification whether input data is positive or negative and we evaluate the precision. Input data is relational tuple in Table 1. Features of retweet actions are (user, "retweet", URL, original tweet id). Features of user actions used for our analyses are those n-tuples, each of them corresponding to one action, including the action type. We made 50.0% negatives for each training dataset, so a random prediction achieves 50.0% precision. We used a machine learning library *Classias*⁶ for this experiment. We adopted *L2 regularized logistic regression* algorithm here. We also conducted same experiments adopting L2 regularized logistic regression, L1 regularized L1 loss SVM, L2 regularized L1 loss SVM, but the following conclusion is same. So we show results only about L2 regularized logistic regression for want of space. 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. The amount of data to predict retweet in $t[2]$ by tweet in $t[1]$ is equal to the amount of data to predict favorite in $t[2]$ by favorite in $t[1]$. When mixing multiple action types as train data, we conducted two experiments. First, we just mixed each data used in the experiments described above. Those experiments are noted as *Not-Fixed* in Table 2. Second we tuned the total amount of data to be equal to the test data. Those experiments are noted as *Fixed* in Table 2.

Results and Discussion

The whole results are shown in Table 2. *Action type for test* means what action type was used for the test experiment, and *Action type for train* means what action type was used for the train experiment. We note tweet, retweet, favorite, following, and Delicious tagging as Tw, RT, Fav, Fol and Tag, respectively in Table 2. In the following we pick up some observations. First, we show results about multi-domain analyses. Then we examine multi-functions analyses on a single domain (Twitter).

⁶www.chokkan.org/software/classias/index.html.en

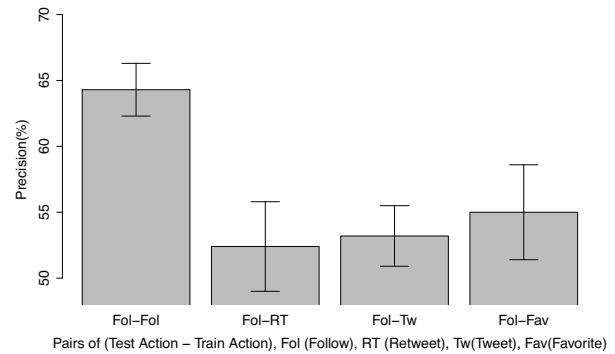


Figure 2: Predicting following by other actions.

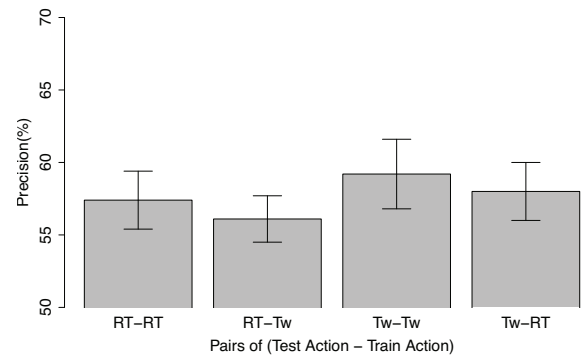


Figure 3: Relations between retweet and tweet.

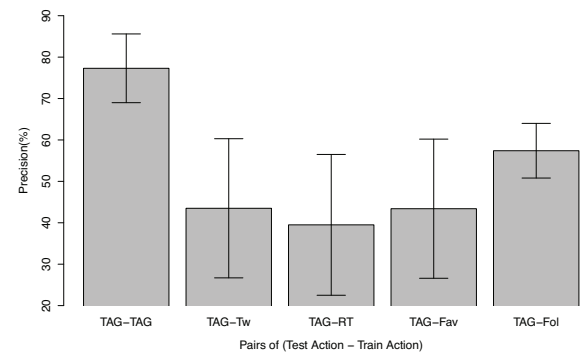


Figure 4: Predicting Delicious tagging by twitter actions.

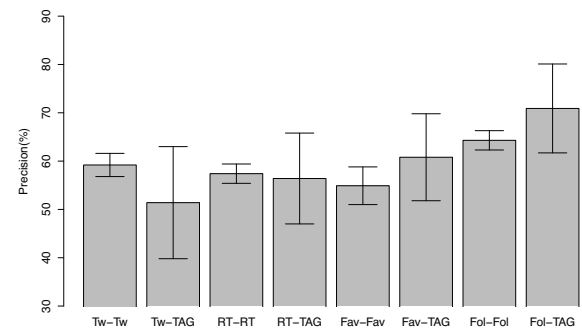


Figure 5: Predicting twitter actions by Delicious tagging.

Table 2: The Precision results (averaged on each time slot) for action types analyses.

Action type for test	Action type for train	Not-Fixed	Fixed
TAG	TAG	77.3 ± 8.3%	77.3 ± 8.3%
TAG	Fav	43.4 ± 1.7%	43.4 ± 1.7%
TAG	Fol	57.4 ± 6.6%	57.4 ± 6.6%
TAG	RT	39.5 ± 17.1%	39.5 ± 17.1%
TAG	Tw	43.5 ± 16.8%	43.5 ± 16.8%
Fav	Fav	54.9 ± 3.9%	54.9 ± 3.9%
Fav	Fol	49.5 ± 2.4%	49.5 ± 2.4%
Fav	RT	53.0 ± 3.3%	53.0 ± 3.3%
Fav	Tw	51.6 ± 2.4%	51.6 ± 2.4%
Fav	TAG	60.8 ± 9.0%	60.8 ± 9.0%
Fav	Tw-Fol	50.5 ± 1.5%	50.3 ± 1.6%
Fav	Fav-Fol	53.3 ± 1.9%	54.3 ± 2.4%
Fav	Fav-RT	55.9 ± 2.9%	55.4 ± 2.9%
Fav	Fav-Tw	55.0 ± 3.1%	54.7 ± 1.8%
Fav	RT-Fol	51.2 ± 1.2%	51.3 ± 1.5%
Fav	RT-Tw	52.5 ± 2.5%	52.6 ± 2.8%
Fav	Fav-RT-Fol	54.7 ± 2.4%	53.9 ± 2.3%
Fav	Fav-RT-Tw	55.3 ± 3.0%	53.8 ± 2.9%
Fav	Fav-Tw-Fol	53.9 ± 1.8%	53.5 ± 3.4%
Fav	RT-Tw-Fol	51.2 ± 1.8%	50.9 ± 1.4%
Fav	Fav-RT-Tw-Fol	54.5 ± 2.2%	52.4 ± 2.8%
Fol	Fav	55.0 ± 3.6%	55.0 ± 3.6%
Fol	Fol	64.3 ± 2.0%	64.3 ± 2.0%
Fol	RT	52.4 ± 3.4%	52.4 ± 3.4%
Fol	Tw	53.2 ± 2.3%	53.2 ± 2.3%
Fol	TAG	70.9 ± 9.2%	70.9 ± 9.2%
Fol	Tw-Fol	62.9 ± 2.3%	62.4 ± 1.3%
Fol	Fav-Fol	63.5 ± 1.8%	62.2 ± 3.2%
Fol	Fav-RT	51.7 ± 2.9%	53.4 ± 2.4%
Fol	Fav-Tw	52.8 ± 2.5%	54.9 ± 2.8%
Fol	RT-Fol	63.2 ± 2.3%	61.9 ± 1.4%
Fol	RT-Tw	52.0 ± 2.8%	53.9 ± 1.9%
Fol	Fav-RT-Fol	62.4 ± 2.2%	59.8 ± 3.2%
Fol	Fav-RT-Tw	51.5 ± 2.8%	53.5 ± 2.8%
Fol	Fav-Tw-Fol	62.2 ± 2.1%	60.2 ± 2.9%
Fol	RT-Tw-Fol	62.2 ± 2.0%	59.6 ± 2.8%
Fol	Fav-RT-Tw-Fol	61.7 ± 1.8%	59.4 ± 3.3%
RT	Fav	53.7 ± 2.6%	53.7 ± 2.6%
RT	Fol	50.8 ± 1.5%	50.8 ± 1.5%
RT	RT	57.4 ± 2.0%	57.4 ± 2.0%
RT	Tw	56.1 ± 1.6%	56.1 ± 1.6%
RT	TAG	56.4 ± 9.4%	56.4 ± 9.4%
RT	Tw-Fol	52.7 ± 1.1%	53.4 ± 1.4%
RT	Fav-Fol	52.0 ± 1.1%	52.1 ± 1.7%
RT	Fav-RT	57.5 ± 2.3%	56.4 ± 4.0%
RT	Fav-Tw	54.6 ± 1.0%	54.5 ± 1.9%
RT	RT-Fol	55.5 ± 1.2%	55.7 ± 1.7%
RT	RT-Tw	58.2 ± 2.1%	57.2 ± 2.3%
RT	Fav-RT-Fol	56.2 ± 1.3%	56.2 ± 2.2%
RT	Fav-RT-Tw	57.9 ± 2.9%	57.7 ± 2.5%
RT	Fav-Tw-Fol	53.1 ± 1.5%	54.2 ± 2.3%
RT	RT-Tw-Fol	56.7 ± 1.9%	56.5 ± 2.1%
RT	Fav-RT-Tw-Fol	56.9 ± 1.9%	55.8 ± 2.0%
Tw	Fav	53.8 ± 3.8%	53.8 ± 3.8%
Tw	Fol	51.0 ± 2.5%	51.0 ± 2.5%
Tw	RT	58.0 ± 2.0%	58.0 ± 2.0%
Tw	Tw	59.2 ± 2.4%	59.2 ± 2.4%
Tw	TAG	51.4 ± 11.6%	51.4 ± 11.6%
Tw	Tw-Fol	56.0 ± 1.7%	54.4 ± 1.3%
Tw	Fav-Fol	51.5 ± 2.5%	50.9 ± 2.4%
Tw	Fav-RT	57.0 ± 2.0%	56.3 ± 2.1%
Tw	Fav-Tw	58.7 ± 2.0%	56.7 ± 2.0%
Tw	RT-Fol	53.7 ± 1.8%	53.7 ± 2.7%
Tw	RT-Tw	59.9 ± 2.2%	58.6 ± 2.2%
Tw	Fav-RT-Fol	53.5 ± 2.2%	53.8 ± 2.0%
Tw	Fav-RT-Tw	59.1 ± 1.7%	58.0 ± 2.0%
Tw	Fav-Tw-Fol	56.0 ± 1.8%	54.7 ± 2.1%
Tw	RT-Tw-Fol	56.6 ± 1.6%	55.6 ± 1.7%
Tw	Fav-RT-Tw-Fol	56.5 ± 1.4%	55.8 ± 1.9%

Complementary features in Multi-Domain Figure 5 shows a result predicting Twitter actions by Delicious tagging and Figure 4 shows a result predicting Delicious tagging by Twitter actions. 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. Delicious tagging actions are effective to predict Twitter favorite actions. When we predict Twitter favorite by using Delicious tagging, it seems to achieve a little higher performances in a mean viewpoint. Delicious tagging actions are also effective to predict Twitter following actions. It seems to achieve almost equal performances to predicting following actions by following itself. This means that Delicious tagging can substitute Twitter 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. This is an asymmetry property of multiple domains. While Delicious tagging was useful to predict some actions in Twitter, actions in Twitter were not so useful to make predictions on Delicious tagging in our experiments. We think one of the reason why we observe this asymmetry is that keywords reflect users’ preferences that complement preferences reflected in actions without keywords. So even if the action types (such as tagging or favorite) are different, adding features such as keywords based on other action types will be effective for interest predictions. This suggestion will support to mix different kind of actions even across web services.

Similar/Dissimilar action types in a single Domain Now we focus attention on a single domain, Twitter. As described in Figure 3, there are some situations in which different action types can be exploited to predict a specific action type. To predict retweet/tweet action, you can achieve almost equal performances by using the other action type, tweet/retweet. Predicting retweet/tweet by the other action type tweet/retweet achieves equal performances and the other action types can substitute the original action type, suggesting that mixing different action types can be effective for sparse data. When one action type B can be exploited to predict another action type A , it will mean that preferences that affect acting A share some characteristics in common with B . We can say an action type B is similar to A . About tweet-retweet relations, it makes sense these action types are similar. Because users can add comments and quote another user’s tweet in their tweets, and this action is similar to clicking a retweet button in the sense that they are both quotations. On the other hand, as described in Figure 2, there are also some situations in which other action types cannot be exploited to predict a specific relation. For example, to predict following actions, using other actions often causes lower performances. When predicting other actions by using following action, it tends to result in lower performances, too. Following is more static than other actions, and following networks can be viewed as platforms for other actions. So the Following actions and other actions may be relatively different in nature.

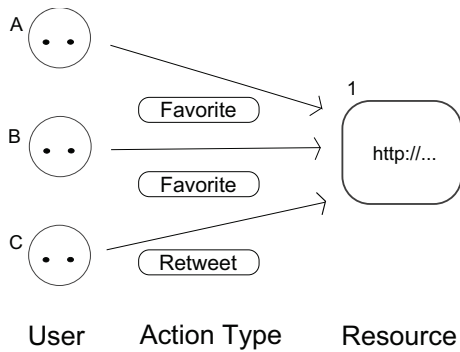


Figure 6: An example of users’ actions on social media.

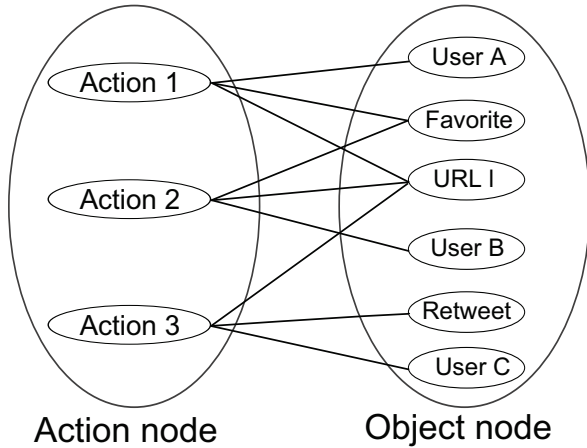


Figure 7: An example of ActionGraph representation.

Ambiguities of the referring URLs actions in a single Domain Compared to the case of following action in Table 2, URLs referring actions in Twitter seem to be more similar with each other. We can say predicting URLs referring by other URLs referring actions is more effective than predicting by using following actions in Twitter. We think this is because that the preferences reflected in URLs referring in Twitter are relatively similar with each other, lapping over each other. This suggests when we can get relatively little data about a specific action type of URLs referring action, other URLs referring actions are expected to substitute it.

The influence of amount of data Comparing the *Fixed* case where we tuned the amount of the total data for train, and the *Not-Fixed* case, we found there are less differences between them. This suggests increases of the amount of data do not necessary lead to linear improvements of performances. Another possible interpretation is that because we collected data from near a specific user’s following network, it may be rather biased. Collecting from more diversified areas might improve the performance. It is one future work to investigate how the properties of the social network affect the prediction.

Proposed Method

We showed adding features based on different action types is effective for the interest prediction task in the previous

section. So it is useful to develop an efficient method that can handle multinomial relations. In this section, we propose a new method that captures both multinomial, user-specific time scale actions performed by users in social media.

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 located by URL *l* with the keyword, “artificial intelligence”. We can view this action as involving four entities: user *A*, URL *l*, the action verb “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 “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} . Note that there are no intra-set edges linking two vertices in V_{AC} or V_{OB} . ActionGraph is a bipartite graph whose edge connects an action node and the object nodes involved in a particular action. Figure 6 shows an example of users’ actions in social media. Figure 7 shows the corresponding ActionGraph.

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 involve 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. Because users are expected to have their own personal time scale (Xiang et al. 2010), it will be effective to preserve the co-occurrences between a user and time in which the user performed the action. On the other hand, if we construct a bipartite graph modeling user-URL co-occurrences, which is popular in recommendation tasks, two important information, entities involved in the action such as an action type and time information for every user, will be lost. It is noteworthy that ActionGraph is not limited to relational triples, and can express co-occurrences among more than three entities. Therefore, we can add other information in ActionGraph that describe a particular action, such as the keywords used for tagging, to better describe a particular action.

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. Such an approach that computes the similarities between a user and a set of items for recommendations is

common practice in recommendation tasks. Following these previous work, 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. 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. Numerous similarity measures have been proposed in the literature to measure the similarity between two sets such as the Jaccard coefficient, Dice coefficient, mutual information etc. Any one of those measures can potentially be used for this purpose.

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. It is worth mentioning that graph kernels consider not only the direct paths between nodes, but also the indirect paths when computing similarity. Moreover, graph kernels have the desirable properties that the similarity between two nodes increases concomitant with the 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 (Eq.1).

$$\mathbf{L}(\mathbf{M}) = \mathbf{D}(\mathbf{M}) - \mathbf{M}. \quad (1)$$

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 (Eq.1) 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,

$$\mathbf{RL}_\beta(\mathbf{AA}^\top) = \sum_{k=0}^{\infty} (-\beta\mathbf{L}(\mathbf{AA}^\top))^k = (\mathbf{I} + \beta\mathbf{L}(\mathbf{AA}^\top))^{-1} \quad (2)$$

where \mathbf{A} is a similarity matrix and \mathbf{I} is the identity matrix, β is a parameter used to tune how much similarity weights

Table 3: 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

are placed on distant nodes when measuring the similarity between a pair of nodes. To explain regularized Laplacian kernel by an example let us consider the situation depicted in Figure 7 where three users, A , B and C express their interest in a URL l . However, action type of A and B is favorite, and the action type of C is retweet. There are more short paths between the action nodes of A and B than between A and C or between B and C . Because the common action type node, favorite, bridges the gap between the action node of A and B , the regularized Laplacian kernel is able to use this fact to accurately compute the similarity. 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.

Evaluations of proposed method via Prediction

Conditions

Dataset The crawling condition is same as the previous analysis. The summary of datasets is in Table 3. *Original-tweet-user* means the user who first tweet the original post and it gets retweeted or favorited by other users.

Prediction setting When a user takes certain actions to a resource, we hypothesized that the user gets interested in the resource. We define the interest prediction problem as predicting resources rank list in which a given user will get interested. The input is n-tuples, each of them corresponding to one action. Retweet action tuple is (user, "retwee", URL, original-tweet-id). The output is similarities between entities. With Twitter dataset, we designed our prediction task as predicting URLs referring actions in retweet or favorite, by exploiting those two action types. We segment the duration into 10 time slots following the previous setting.

Evaluation metrics We use two types of evaluation metrics. First we evaluate the degree of accuracy and then we evaluate the coverage, because coverage is effective to evaluate the robustness against data sparseness. For accuracy evaluations, we use *R-Precision* metric, which is adopted in Recommendation Task and Information Retrieval. 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. For instance, one may refer thirty URLs, and another may refer only two URLs in three days. So R-Precision will be suitable for our measure. We averaged R-Precision for each user and each slot. For coverage evaluations, we define *Coverage* as the percentage of (user, URL) pairs in the test set for which we can compute a recommendation. This information helps to make recommendations for cold start users.

Compared methods We adopt two methods as compared methods; one is a standard method for dealing with multinomial relations, and the other is one of the state-of-the-art methods for recommendation tasks. (1) Standard tensor analysis (PARAFAC) (Bader and Kolda 2006). A tensor is a mathematical representation of a multi-way array. Tensor decomposition or factorization is a form of higher-order principal component analysis, and PARAFAC decomposition is one of the most popular tensor decompositions. 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. (2) latent Dirichlet allocation (LDA)-based Collaborative Filtering. 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.

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 compared methods. In our prediction method, we changed the parameter β in in the equation 2 as 0.01, 0.05, 0.01. 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. In PARAFAC, we changed the reduced latent dimension as 200, 400, 600, 800. The result was best when adopting the 600 dimension, so we fixed it as 600. In LDA, we changed the topic number as 100, 200, 400, 600, and got 400 as the best parameter. 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 get 0.01. It is pointed out that changing α by the topic number is effective (Wallach, Mimno, and Mccallum 2010), so we set the α for each topic as $\frac{50}{\text{topic-number}}$ following the pLDA⁷ implementation we used.

Results and Discussion

The results (mean and standard deviation) are given in Table 4. In R-Precision, ActionGraph outperforms both PARAFAC and LDA. The reason why ActionGraph outperformed LDA, which is a state-of-the-art method for binomial relations, can be thought that multinomial information is indeed important to predict user social actions. In Coverage, ActionGraph significantly outperforms PARAFAC. ActionGraph can compute almost all similarities for possible pairs. When modeling multi-dimensional data as a tensor, data sparseness is one of the biggest problems, and actually PARAFAC’s Coverage is poor compared to our methods.

⁷www.code.google.com/p/plda

Table 4: The average prediction performances by R-Precision and Coverage (averaged on each user and each time slot) for evaluations via prediction.

	ActionGraph	PARAFAC	LDA
R-Precision	7.6±3.3 %	3.4±2.1 %	4.3±1.4 %
Coverage	99.8±0.0 %	43.6±6.7 %	99.0±0.2 %

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.

Related Work

Our proposed model can capture two properties of social media, multinomial relation and time-evolving user-specific time scale. First we review multinomial relational analyses work and then review some work dealing with time.

Multinomial Relation Analysis

By multinomial relation analyses, the correlation of more than three entities can be captured. This is useful in many situations. For example, even if two users tagged the same keyword to the same document, the meaning of the tag can be different for each user. If you preserve the correlation 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. As relations become higher order, the combinations of data increase exponentially. These approaches are promising, but they are not explicitly modeling user-specific time dimensions.

Time-Evolving Network Analysis

There are works (Sun et al. 2007; Sun, Tao, and Faloutsos 2006) that handle time as a 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 tagging them. So time-evolving approaches should be able to handle multi-dimensional data.

Conclusion

By analyzing relations among various actions in social media, we got suggestions that will be useful for aggregating, integrating diverse data from multiple web services, multiple functions provided on the Web. In addition, we proposed ActionGraph, a novel graph representation for modeling users' multinomial, time-evolving actions. We showed our method outperforms standard tensor-based analysis and existing state-of-the-art for recommendations in prediction tasks. Our method has both higher precision and higher coverage, and expected to be effective for sparse data. We hope this work can contribute to think, create the future of social media.

Acknowledgments

Thank Kiyohiro YAMAGUCHI, The University of Tokyo, for his help for this research.

References

- Bader, B. W., and Kolda, T. G. 2006. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. In *TOMS* 32(4), 635–653.
- Basilico, J., and Hofmann, T. 2004. Unifying collaborative and content-based filtering. In *Proc. 21st International Conference on Machine Learning (ICML 2004)*.
- Ben-Hur, A., and Noble, W. S. 2005. Kernel methods for predicting protein-protein interactions. In *Proc. 13th International Conference on Intelligent Systems for Molecular Biology. Bioinformatics* 21, i38–i46.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.
- Fouss, F.; Francoise, K.; Yen, L.; Pirotte, A.; and Saerens, M. 2006. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proc. 6th International Conference on Data Mining (ICDM 2006)*, 863–868.
- Ito, T.; Shimbo, M.; Kudo, T.; and Matsumoto, Y. 2005. Application of kernels to link analysis. In *Proc. 11th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2005)*, 586–592.
- Jamali, M., and Ester, M. 2009. 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)*.
- Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. *Advances Neural Information Processing Systems* 13:556–562.
- Lin, Y.-R.; Sun, J.; Castro, P.; Konuru, R.; Sundaram, H.; and Kelliher, A. 2009. Metafac: community discovery via relational hypergraph factorization. In *Proc. 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009)*.
- Massa, P., and Bhattacharjee, B. 2005. Using trust in recommender systems: An experimental analysis. In *International Conference on Web Intelligence (WI 2005)*, 221–235.
- Matsuo, Y., and Yamamoto, H. 2009. Community gravity: Measuring bidirectional effects by trust and rating on online social networks. In *Proc. 18th International World Wide Web Conference (WWW 2009)*.
- Oyama, S., and Manning, C. D. 2004. Using feature conjunctions across examples for learning pairwise classifiers. In *Proc. 15th European Conference on Machine Learning (ECML 2004)*, 322–333.
- Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of The Conf. on Computer Supported Cooperative Work*.
- Shardanand, U., and Maes, P. 1995. Social information filtering: Algorithms for automating gword of mouth. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*.
- Smola, A. J., and Kondor, R. 2003. *Kernels and regularization on graphs*. Springer.
- Sun, J.; Faloutsos, C.; Papadimitriou, S.; and Yu, P. S. 2007. Graphscope: parameter-free mining of large time-evolving graphs. In *Proc. 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2007)*, 687–696.
- Sun, J.; Tao, D.; and Faloutsos, C. 2006. Beyond streams and graphs: dynamic tensor analysis. In *Proc. 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2006)*, 374–383.
- Wallach, H. M.; Mimno, D.; and McCallum, A. 2010. Rethinking lda: Why priors matter. In *In Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS 2009)*.
- Xiang, L.; Yuan, Q.; Zhao, S.; Chen, L.; Zhang, X.; Yang, Q.; and Sun, J. 2010. 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)*.
- Zhou, D.; Zhu, S.; Yu, K.; Song, X.; Tseng, B. L.; Zha, H.; and Giles, C. L. 2008. Learning multiple graphs for document recommendations. In *Proc. 17th International World Wide Web Conference (WWW 2008)*, 141–150.