Semantic Data Representation for Improving Tensor Factorization

Makoto Nakatsuji[1], Yasuhiro Fujiwara[2], Hiroyuki Toda[3], Hiroshi Sawada[4], Jin Zheng[5], James A. Hendler[6]

[1,3,4]NTT Service Evolution Laboratories, NTT Corporation, 1-1 Hikarinooka, Yokosuka-Shi, Kanagawa 239-0847 Japan
[2]NTT Software Innovation Center, NTT Corporation, 3-9-11 Midori-Cho, Musashino-Shi, Tokyo 180-8585 Japan
[5,6]Tetherless World Constellation, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180-3590 USA
[1,2,3,4]{nakatsuji.makoto, fujiwara.yasuhiro, toda.hiroyuki, sawada.hiroshi}@lab.ntt.co.jp
[5]zhengj3@rpi.edu, [6]hendler@cs.rpi.edu

Abstract

Predicting human activities is important for improving recommender systems or analyzing social relationships among users. Those human activities are usually represented as multi-object relationships (e.g. user's tagging activities for items or user's tweeting activities at some locations). Since multi-object relationships are naturally represented as a tensor, tensor factorization is becoming more important for predicting users' possible activities. However, its prediction accuracy is weak for ambiguous and/or sparsely observed objects. Our solution, Semantic data Representation for Tensor Factorization (SRTF), tackles these problems by incorporating semantics into tensor factorization based on the following ideas: (1) It first links objects to vocabularies/taxonomies and resolves the ambiguity caused by objects that can be used for multiple purposes. (2) It next links objects to composite classes that merge classes in different kinds of vocabularies/taxonomies (e.g. classes in vocabularies for movie genres and those for directors) to avoid low prediction accuracy caused by rough-grained semantics. (3) It then lifts sparsely observed objects into their classes to solve the sparsity problem for rarely observed objects. To the best of our knowledge, this is the first study that leverages semantics to inject expert knowledge into tensor factorization. Experiments show that SRTF achieves up to 10% higher accuracy than state-of-the-art methods.

Introduction

Analyzing multi-object relationships, formed by three or more kinds of objects, is critical for predicting human activities in detail. Typical relationships include those among users, items, and user-assigned tags against items within content providing services. They also include relationships among users, their tweets, and their locations as posted in twitter, etc. Those prediction results can improve many applications such as recommendations and social network analysis. For example, suppose that a user highly rates a thriller movie with tag "romance" and another user also highly rates the same title with tag "car action". Methods that only handle bi-relational (user-item) objects ignore tags, and finds above users to be highly similar because they share the same movies. Multi-relational methods compute those users as similar but also slightly different because they have different opinions about the item. Recommendation quality is improved by reflecting those differences in detail as (Nakatsuji and Fujiwara 2014) described.

Representing multi-object relationships by a tensor is natural, so predicting possible future relationships by tensor factorization is important (Karatzoglou et al. 2010). Among known proposals, Bayesian Probabilistic Tensor Factorization (BPTF) (Xiong et al. 2010) is promising because of its efficient sampling of large-scale datasets with easy parameter settings. Present tensor factorization schemes, however, have weak accuracy because they fail to utilize the semantics underlying the objects and to handle objects that are ambiguous and/or sparsely observed.

Semantic ambiguity is one of the fundamental problems in text clustering. Several researches use WordNet (Miller 1995) or Wikipedia to solve semantic ambiguity and improve performance in text clustering (Hu et al. 2008) or computing the semantic relatedness of documents (Gabrilovich and Markovitch 2007). Recent semantic web studies also pointed out that the accuracy of opinion mining or recommendation can fall if they use *rough-grained semantics* to capture users' interests (Nakatsuji et al. 2012; Saif, He, and Alani 2012; Nakatsuji and Fujiwara 2014). (Parundekar, Knoblock, and Ambite 2012) creates composite classes that merge classes in different kinds of vocabularies/taxonomies (e.g. class "populated place" in geoNames (Jain et al. 2010) and class "city" in DBPedia (Bizer et al. 2009)) to find more complete definitions using Linked Open Data (LOD) (Bizer, Heath, and Berners-Lee 2009). We also consider that fine-grained semantics improves tensor factorization accuracy. Solving *the sparsity problem* is also an urgent goal given that most training datasets are not sufficient (Narita et al. 2011). To improve tensor factorization accuracy, (Narita et al. 2011; Ermis, Acar, and Cemgil 2012; Takeuchi et al. 2013) use auxiliary information in addition to multi-object relationships. There are, however, no published tensor methods that use the taxonomy of objects even though taxonomies are easily available for real applications due to tools like DBPedia or other linked data.

This paper proposes a method, Semantic data Representation for Tensor Factorization (SRTF), that incorporates semantics into tensor factorization by using the following three ideas: (1) It first measures the similarities between objects and

instances in vocabularies/taxonomies and then links objects to instances with the highest similarity values to resolve object ambiguity. Fig. 1 represents an example of the relationships among three objects; users, items, and tags. In Fig. 1-(1), user u_m assigns tag "Breathless" to movie item v_n whose name is "Rock". There are, however, several movie items whose name is "Rock". SRTF measures similarity between metadata given for v_n and properties given for instances in Freebase or DBPedia. Then, it links v_n to the instance "Rock" in class "$C3$: Action" if it is the most similar instance with item v_n. SRTF also resolves the ambiguity caused by tags like "Breathless" in Fig. 1. (2) It next creates composite classes (Parundekar, Knoblock, and Ambite 2012) by computing disjunctions of conjunctions of instances among classes in different kinds of vocabularies. SRTF links objects to instances in composite classes rather than separate classes. This enables us to analyze multi-object relationships in a detailed semantic space. In Fig. 1-(2), items linked to the instances "Rock" or "Island" are included in class "Action" in a genre vocabulary and "Michael Bay" in a director vocabulary. SRTF creates the composite class "$C4$: Action&Michael Bay" and links v_n to instance "Rock" in $C4$ in addition to "Rock" in $C3$. (3) It then lifts objects linked to LOD/WordNet instances to classes if those objects are sparsely observed. SRTF then applies semantic biases to sparse objects in tensor factorization by using shared knowledge in classes. Fig. 1-(3) assumes that there are only a few observations for items linked to "Rock" and "Island", and tags linked to "Breathtaking" and "Breathless". In such a case, normal factorized results tend not to reflect such sparsely observed objects, which decreases the prediction accuracy. SRTF applies semantic biases from class $C3$ and composite class $C4$ (or class $C1$) to those sparse objects in tensor factorization. This well solves the sparsity problem.

SRTF creates a disambiguated tensor by linking objects to classes and augmented tensors by lifting sparse objects to classes. It then factorizes those tensors over the BPTF framework simultaneously. As a result, it can incorporate semantic biases, computed by factorizing augmented tensors, into feature vectors for sparse objects during the factorization process. Thus, it solves the sparsity problem.

We applied SRTF to predict users' rating activities in this paper and evaluated SRTF using (1) the MovieLens ratings/tags[1] with FreeBase (Bollacker et al. 2008)/WordNet and (2) the Yelp ratings/reviews[2] with DBPedia (Bizer et al. 2009). The results show that SRTF achieves up to 10% higher accuracy than state-of-the-art methods.

Related works

Tensor factorization methods have recently been used in various applications such as recommendation (Karatzoglou et al. 2010; Rendle and Schmidt-Thieme 2010) and social network analysis (Zheng et al.). Recently, efficient tensor factorization method based on a probabilistic framework has appeared (Xiong et al. 2010). Unfortunately, one major problem with tensor factorization is that its prediction accuracy often tends to be poor because observations in real datasets are typically sparse (Narita et al. 2011). Generalized coupled tensor factorization (GCTF) (Yilmaz, Cemgil, and Simsekli 2011; Ermis, Acar, and Cemgil 2012; 2013) and a few other studies (Zheng et al. ; Acar, Kolda, and Dunlavy 2011; Takeuchi et al. 2013) try to incorporate extra information into tensor factorization by simultaneously factorizing observed tensors and matrices representing extra information. Taxonomy is now also used for clustering genomes with functional annotations in genomic science (Nakaya et al. 2013). There are, however, no tensor studies that focus on handling ambiguous and sparse objects by using vocabularies/taxonomies of objects.

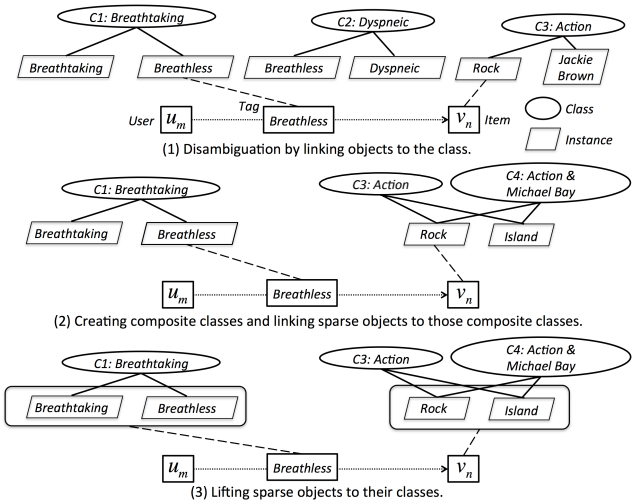

Figure 1: Linking objects to classes by three ideas.

Preliminary

We now explain the background techniques of the paper.

Vocabularies and taxonomies Vocabularies/taxonomies, sometimes called "simple ontologies" (McGuinness 2003), are collections of human-defined classes and usually have a hierarchical structure either as a graph (vocabulary) or a tree (taxonomy). This paper uses, in particular, the DBPedia and Freebase vocabularies and the WordNet taxonomy to improve tensor factorization accuracy. DBPedia and Freebase have many item entries with many properties (e.g. genres or directors) such as music and movies; we use item entries as instances and entries indicated by some specific properties (e.g genre property) from item entries as classes. WordNet is a lexical database for the English language and is commonly exploited to support automatic analysis of texts. In WordNet, each word is classified into one or more synsets, each of which represents a class; we consider a synset as a class and a word classified in the synset as an instance.

Bayesian Probabilistic Tensor Factorization This paper deals with the multi-object relationships formed by user u_m, item v_n, and tag t_k. Number of users is M, that of items is N, and that of tags is K. Third-order tensor $\mathcal{R}$ models the relationships among objects from sets of users, items, and tags. The (m,n,k)-th element $r_{m,n,k}$ indicates the m-th

[1] Available at http://www.grouplens.org/node/73

[2] Available at http://www.yelp.com/dataset_challenge/

user's rating of the n-th item with the k-th tag. Tensor factorization assigns a D-dimensional latent feature vector to each user, item, and tag, denoted as $\mathbf{u}_m$, $\mathbf{v}_n$, and $\mathbf{t}_k$, respectively. Accordingly, $\mathcal{R}$ can be approximated as follows:

$$r_{m,n,k} \approx \langle \mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k \rangle \equiv \sum_{i=1}^{D} u_{m,i} \cdot v_{n,i} \cdot t_{k,i} \qquad (1)$$

where index i represents the i-th element of each vector.

BPTF (Xiong et al. 2010) puts tensor factorization into a Bayesian framework by assuming a generative probabilistic model for ratings with Gaussian/Wishart priors (Bishop 2006) over parameters. We denote the matrix representation of $\mathbf{u}_m$, $\mathbf{v}_n$, and $\mathbf{t}_k$ as $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{T}$, respectively. The size of $\mathbf{U}$ is $M \times D$, that of $\mathbf{V}$ is $N \times D$, and that of $\mathbf{T}$ is $K \times D$. To account for randomness in ratings, BPTF uses the below probabilistic model for generating ratings: $r_{m,n,k}|\mathbf{U},\mathbf{V},\mathbf{T} \sim \mathcal{N}(\langle \mathbf{u_m}, \mathbf{v_n}, \mathbf{t_k} \rangle, \alpha^{-1})$. This equation represents the conditional distribution of $r_{m,n,k}$ given $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{T}$ as a Gaussian distribution with mean $\langle \mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k \rangle$ and precision α.

The generative process of BPTF is:

1. Generate $\Lambda_\mathbf{U}$, $\Lambda_\mathbf{V}$, and $\Lambda_\mathbf{T} \sim \mathcal{W}(\mathbf{W}_0, \nu_0)$, where $\Lambda_\mathbf{U}$, $\Lambda_\mathbf{V}$, and $\Lambda_\mathbf{T}$ are the precision matrices for Gaussians. $\mathcal{W}(\mathbf{W}_0, \nu_0)$ is the Wishart distribution with ν_0 degrees of freedom and a $D \times D$ scale matrix, $\mathbf{W}_0$: $\mathcal{W}(\Lambda|\mathbf{W}_0, \nu_0) = \frac{|\Lambda|^{(\nu_0 - D - 1)/2}}{C} \exp(-\frac{Tr(\mathbf{W}_0{}^{-1}\Lambda)}{2})$ where C is a constant.
2. Generate $\mu_\mathbf{U} \sim \mathcal{N}(\mu_0, (\beta_0 \Lambda_\mathbf{U})^{-1})$, $\mu_\mathbf{V} \sim \mathcal{N}(\mu_0, (\beta_0 \Lambda_\mathbf{V})^{-1})$, and $\mu_\mathbf{T} \sim \mathcal{N}(\mu_0, (\beta_0 \Lambda_\mathbf{T})^{-1})$, where $\mu_\mathbf{U}$, $\mu_\mathbf{V}$, and $\mu_\mathbf{T}$ are used as the mean matrices for Gaussians.
3. Generate $\alpha \sim \mathcal{W}(\tilde{W}_0, \tilde{\nu}_0)$.
4. For each $m \in (1 \ldots M)$, generate $\mathbf{u}_m \sim \mathcal{N}(\mu_\mathbf{U}, \Lambda_\mathbf{U}{}^{-1})$.
5. For each $n \in (1 \ldots N)$, generate $\mathbf{v}_n \sim \mathcal{N}(\mu_\mathbf{V}, \Lambda_\mathbf{V}{}^{-1})$.
6. For each $k \in (1 \ldots K)$, generate $\mathbf{t}_k \sim \mathcal{N}(\mu_\mathbf{T}, \Lambda_\mathbf{T}{}^{-1})$.
7. For each non-missing entry (m, n, k), generate $r_{m,n,k} \sim \mathcal{N}(\langle \mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k \rangle, \alpha^{-1})$.

Parameters μ_0, β_0, $\mathbf{W}_0$, ν_0, $\tilde{W}_0$, and $\tilde{\nu}_0$ are set properly for the objective dataset; varying their values, however, has little impact on the final prediction (Xiong et al. 2010).

BPTF views the hyper-parameters α , $\Theta_\mathbf{U} \equiv \{\mu_\mathbf{U}, \Lambda_\mathbf{U}\}$, $\Theta_\mathbf{V} \equiv \{\mu_\mathbf{V}, \Lambda_\mathbf{V}\}$, and $\Theta_\mathbf{T} \equiv \{\mu_\mathbf{T}, \Lambda_\mathbf{T}\}$ as random variables, yielding a predictive distribution for unobserved rating $\hat{r}_{m,n,k}$, which, given observable tensor $\mathcal{R}$, is:

$$p(\hat{r}_{m,n,k}|\mathcal{R}) = \int p(\hat{r}_{m,n,k}|\mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k, \alpha) \; p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_\mathbf{U}, \Theta_\mathbf{V}, \Theta_\mathbf{T}|\mathcal{R}) \; d\{\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_\mathbf{U}, \Theta_\mathbf{V}, \Theta_\mathbf{T}\}. \qquad (2)$$

BPTF views Eq. (2) as the expectation of $p(\hat{r}_{m,n,k}|\mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k, \alpha)$ over the posterior distribution $p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_\mathbf{U}, \Theta_\mathbf{V}, \Theta_\mathbf{T}|\mathcal{R})$, and approximates the expectation by an average of samples drawn from the posterior distribution. Since the posterior is too complex to directly sample from, the indirect sampling technique of Markov Chain Monte Carlo (MCMC) is invoked.

The time and space complexity of BPTF is $O(\#nz \times D^2 + (M+N+K) \times D^3)$ where $\#nz$ is the number of observations. M, N, or K is greater than D. Typically, the first term is much larger than the rest. So, it is simpler than typical tensor methods (e.g. GCTF requires $O(M \times N \times K \times D)$ as described in (Ermis, Acar, and Cemgil 2012)). BPTF computes feature vectors in parallel while avoiding fine parameter tuning. To initialize the sampling, it adopts *maximum a posteriori* (MAP) results from Probabilistic Matrix Factorization (PMF) (Salakhutdinov and Mnih 2008b); this lets the MCMC converge quickly. Accordingly, we base our method on the BPTF framework.

Method

We describe our method in detail in this subsection.

Disambiguation

This process has the following three parts:

(1) Disambiguation for items Content providers often use specific, but simple, vocabularies to manage items. This degrades abstraction process (explained in next section) after disambiguation process because such simple vocabularies can express only rough-grained semantics for items. Thus, our solution is to link items to instances in Freebase or DBPedia vocabularies and resolve ambiguous items. For example, in our evaluation, the MovieLens original item vocabulary has only 19 classes while that of Freebase has 724 composite classes. These additional terms are critical to the success of our method.

We now explain the procedures of the disambiguation process of SRTF: (i) It creates a property vector for item v_n, $\mathbf{p}_{v_n}$; x-th element of $\mathbf{p}_{v_n}$ has the value for corresponding property p_x. It also creates a property vector for instance e_j in Freebase/DBPedia, $\mathbf{p}_{e_j}$, which has the same properties as $\mathbf{p}_{v_n}$. (ii) SRTF then computes the cosine similarity of property vectors and links item v_n to instance $v_{n'}$ which has the highest similarity value with v_n. Because items with the same name can be disambiguated to different item instances, the number of items, N, and the number of item instances after disambiguation, N', can be different values; $N \leq N'$. (iii) After linking the item to instance $v_{n'}$, SRTF can use the Freebase/DBPedia vocabulary (e.g. genre vocabulary) to identify classes of $v_{n'}$.

We express the above process by function $f(v_n)$; if v_n can be disambiguated to instance $v_{n'}$, $f(v_n)$ returns class set that has $v_{n'}$. Otherwise, it returns empty set $\emptyset$.

(2) Disambiguation for tags/reviews Many tags are ambiguous as explained in Fig. 1 because they are assigned by users freely. Thus, we need to disambiguate tags by using WordNet to improve prediction accuracy. SRTF first classifies tags into those representing content of items or the subjectivity of users for items as (Cantador, Konstas, and Jose 2011; Nakatsuji and Fujiwara 2014) indicate that those tags are useful in improving prediction accuracy. When determining content tags, SRTF extracts noun phrases from tags because the content tags are usually nouns (Cantador, Konstas, and Jose 2011). First, SRTF removes some stop-words (e.g. conjunctions), then determines the phrases as tuples of PoS (Part of Speeches), and finally

compare the resulting set with the following set of POS-tuple patterns defined for classifying content tags: [<noun>], [<adjective><noun>], and [<determiner><noun>]. It also compares the resulting set with the following set of POS-tuple patterns defined for classifying subjective tags: [<adjective>], [<adjective><noun>], [<adverb>], [<adverb><adjective>], and [*<pronoun>*<adjective>*]. It also examines negative forms of patterns using the Stanford-parser (Klein and Manning 2003) to identify tags like "Not good".

Then, SRTF links the tags to the WordNet taxonomy. It analyzes properties assigned to item v_n in linking tag t_k to the taxonomy because t_k often reflects v_n's characteristics. We apply a semantic similarity measurement method (Zheng et al. 2013) based on VSM (Turney and Pantel 2010) for disambiguation, and use it as follows: (i) It first crawls the descriptions of WordNet instances associated with word w in tag t_k. w is noun if t_k is content tag. w is adjective or adverb if t_k is subjective tag. Each WordNet instance w_j has description d_j. (ii) It next removes some stop-words from description d_j and constructs vector $\mathbf{w_j}$ whose elements are words in d_j and values are observed counts of corresponding words. (iii) It also crawls the description of item instance $v_{n'}$ and descriptions of genres of $v_{n'}$ from Freebase/DBPedia. It constructs vector $\mathbf{i_{n'}}$ whose elements are words and values are observed counts of corresponding words in the descriptions. Thus, vector $\mathbf{i_{n'}}$ represents the characteristics of $v_{n'}$ and its genres. (iv) Finally, it computes the cosine similarity of $\mathbf{i_{n'}}$ and $\mathbf{w_j}$, then links t_k to the wordNet instance w_j that has the highest similarity value with t_k and considers w_j as $t_{k'}$. Because tags with the same name can be disambiguated to different WordNet instances, the number of tags, K, and the number of WordNet instances after disambiguation, K', can be different values; $K \leq K'$.

This paper also applies SRTF to multi-object relationships among users, items, and user-assigned reviews. In this case, we replace the reviews with aspect/subjective phrases in reviews, treat those phrases as tags, and input replaced relationships to SRTF. This is because aspect/subjective tags reflect users' opinions of an item (Yu et al. 2011). Of particular note, we use the aspect tags extracted by using a semantics-based mining study (Nakatsuji, Yoshida, and Ishida 2009). It analyzes reviews for specific objective domains (e.g. music) and extracts aspect tags from reviews that match the instances (e.g. artists) in the vocabulary for that domain. Thus, extracted tags are already linked to the vocabulary.

We express the above process by function $g(v_n, t_k)$; if t_k can be disambiguated to $t_{k'}$, $g(v_n, t_k)$ returns class that has wordNet instance $t_{k'}$. Otherwise, it returns empty set $\emptyset$.

(3) Constructing the disambiguated tensor SRTF constructs the disambiguated tensor $\mathcal{R}^d$ as follows: (i) It picks up each observation $r_{m,n,k}$ in $\mathcal{R}$. It disambiguates item v_n (tag t_k) in each observation by linking v_n (t_k) to instance $v_{n'}$ ($t_{k'}$) in the vocabulary/taxonomy. (ii) It replaces observation $r_{m,n,k}$ in $\mathcal{R}$ to observation $r^d_{m,j,l}$ in $\mathcal{R}^d$ as follows:

$$r^d_{m,j,l} = \begin{cases} r_{m,n,k} & ((f(v_n) = \emptyset)) \\ r_{m,n',k} & ((f(v_n) \neq \emptyset) \cap (g(v_n, t_k) = \emptyset)) \\ r_{m,n',k'} & ((f(v_n) \neq \emptyset) \cap (g(v_n, t_k) \neq \emptyset)) \end{cases} \quad (3)$$

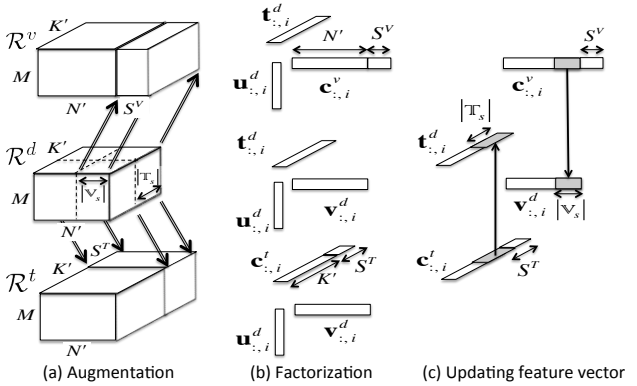

Figure 2: Examples of our factorization process.

(iii) It repeats steps (i) to (ii) by changing the entry in $\mathcal{R}$. As a result, it obtains the disambiguated tensor $\mathcal{R}^d$.

This process maps $\mathcal{R}$ to a semantic concept space and allows us also to conduct abstraction as described next.

Abstraction

We create augmented tensors by lifting sparse instances in $\mathcal{R}^d$ to their classes. This allows us to analyze the shared knowledge in classes in tensor factorization. Thus, it overcomes the sparsity problem. It has the following two parts:

(1) Abstraction by taxonomy/vocabulary classes First, we define the set of sparse item instances, $\mathbb{V}_s$. This is defined as the group of the most sparsely observed items v_ss among all items. We set a 0/1 flag to indicate the observation existence of multi-object relationships composed by user u_m, item $v_{n'}$, and tag $t_{k'}$ in $\mathcal{R}^d$ as $o^d_{m,n',k'}$. Then, $\mathbb{V}_s$ is computed as $\mathbb{V}_s = \{v_s : (\sum_{r_{m,s,k'} \in \mathcal{R}^d} o^d_{m,s,k'} / \sum_{r_{m,n',k'} \in \mathcal{R}^d} o^d_{m,n',k'}) < \delta\}$.

Here, δ is the parameter used to determine the number of sparse instances in $\mathbb{V}_s$. Typically, we set δ ranged from 0.1 to 0.3 according to the long tail characteristic (Anderson 2006). We denote a class of sparse instances as s^v_s and the number of classes of sparse instances as $S^V = |\bigcup_{\mathbb{V}_s} f(v_s)|$. Then, SRTF constructs augmented tensor for items $\mathcal{R}^v$ by inserting multi-object relationships composed by users, classes of sparse item instances, and tag instances into $\mathcal{R}^d$ as:

$$r^v_{m,j,k'} = \begin{cases} r^d_{m,j,k'} (j \leq N') \\ r^d_{m,s,k'} (N' < j \leq (N' + S^V)) \cap (s^v_{(j-N')} \in f(v_s)) \end{cases} \quad (4)$$

The semantically augmented tensor for tags $\mathcal{R}^t$ can be computed in the same way as $\mathcal{R}^v$.

In Fig. 2-(a), SRTF lifts items in the sparse item set $\mathbb{V}_s$ to their classes (the size is S^V) and creates an augmented tensor $\mathcal{R}^v$ from the disambiguated tensor $\mathcal{R}^d$. It also lifts tags in the sparse tag set $\mathbb{T}_s$ to their classes (the size is S^T) and creates an augmented tensor $\mathcal{R}^t$ from $\mathcal{R}^d$.

(2) Abstraction by composite classes SRTF also creates composite classes (Parundekar, Knoblock, and Ambite 2012), defined as disjunctions of conjunctions of "property and value" combinations assigned for item instances, to capture detailed semantics.

We define the class, restricted by property p_x, as c^x. Then, we define a composite class that has X kinds of properties by its populated instances. A set of item instances in composite class c' is computed as $\{v_j : v_j \in \bigcap_{1 \leq x \leq X} \{v_{n'} \in c^x\}\}$. Then, we define function $f'(v_n)$; if v_n can be linked to $v_{n'}$, it returns a set of composite classes that has $v_{n'}$. Otherwise, it returns $\emptyset$. SRTF uses $f'(v_n)$ in spite of $f(v_n)$ in Eq. (4) and additionally inserts multi-object relationships composed by users, composite classes of sparse items, and tags into $\mathcal{R}^v$.

As a result, SRTF can analyze multi-object relationships over fine-grained semantic space in tensor factorization.

Tensor factorization with semantic biases

SRTF computes predictions by factorizing disambiguated tensor $\mathcal{R}^d$ and augmented tensors $\mathcal{R}^v$ and $\mathcal{R}^t$, simultaneously, in the BPTF framework. Please see Fig. 2. For ease of understanding, the figure factorizes tensors into row vectors of matrices $\mathbf{U}^d$, $\mathbf{V}^d$, and $\mathbf{T}^d$. i-th row vectors of $\mathbf{U}^d$, $\mathbf{V}^d$, and $\mathbf{T}^d$ are denoted as $\mathbf{u}^d_{:,i}$, $\mathbf{v}^d_{:,i}$, and $\mathbf{t}^d_{:,i}$, respectively ($1 \leq i \leq D$). SRTF gives semantic knowledge in $\mathbf{c}^v_j$ and $\mathbf{c}^t_j$ to feature vectors $\mathbf{v}_{n'}$ and $\mathbf{t}_{k'}$ if $v_{n'}$ and $t_{k'}$ are sparsely observed. It also circulates the semantic knowledge via shared parameters (e.g. $\mathbf{v}_{n'}$ and $\mathbf{t}_{k'}$) during factorizations of the three tensors. Thus, the semantic knowledge can effectively be distributed among sparse objects during tensor factorization. It greatly solves the sparsity problem.

Approach We first explain the following key approaches underlying our factorization method:

(A) SRTF factorizes tensors $\mathcal{R}^d$, $\mathcal{R}^v$, and $\mathcal{R}^t$ simultaneously. It computes semantically-biased feature vectors $\mathbf{c}^v_j$ and $\mathbf{c}^t_j$ by factorizing $\mathcal{R}^v$ and $\mathcal{R}^t$ as well as feature vectors $\mathbf{v}^d_{n'}$ and $\mathbf{t}^d_{k'}$ by factorizing $\mathcal{R}^d$. Fig. 2-(b) represents the i-th row vector of feature vectors factorized from tensors.

(B) SRTF shares precision α, feature vectors $\mathbf{u}^d_m$, $\mathbf{v}^d_{n'}$ and $\mathbf{t}^d_{k'}$, and their hyperparameters, which are computed by factorizing $\mathcal{R}^d$, in the factorizations of $\mathcal{R}^v$ and $\mathcal{R}^t$. Thus, semantic biases can be shared among the three tensors via those shared parameters. In Fig 2-(b), $\mathbf{u}^d_{:,i}$ and $\mathbf{t}^d_{:,i}$ are shared in factorizing augmented tensor $\mathcal{R}^v$. Similarly, $\mathbf{u}^d_{:,i}$ and $\mathbf{v}^d_{:,i}$ are shared in factorizing augmented tensor $\mathcal{R}^t$.

(C) SRTF updates latent feature $\mathbf{v}^d_{n'}$ to $\mathbf{v}'_{n'}$ by incorporating semantic biases into $\mathbf{v}^d_{n'}$. In Fig. 2-(c), each row vector $\mathbf{c}^v_{:,i}$ has latent features for N' items and those for S^V classes. The features in $\mathbf{c}^v_{:,i}$ share semantic knowledge of sparse items. Thus, they are useful to solve the sparsity problem caused by those sparse items. So, SRTF replaces latent feature $\mathbf{v}^d_{n'}$ with $\mathbf{c}^v_{n'}$ if $v_{n'}$ is a sparse item instance. In the same way, SRTF updates tag feature vector $\mathbf{t}^d_{k'}$ to $\mathbf{t}'_{k'}$ by incorporating semantic biases into $\mathbf{t}^d_{k'}$ if $t_{k'}$ is a sparse tag instance.

Implementation over the BPTF framework SRTF applies $\mathbf{u}^d_m$, $\mathbf{v}'_{n'}$, and $\mathbf{t}'_{k'}$, and their hyperparameters to Eq. (2). It learns the predictive distribution for unobserved ratings $\hat{r}^d_{m,n',k'}$ by MCMC procedure as per BPTF. Detailed MCMC procedure of BPTF can be seen in (Xiong et al. 2010). MCMC collects a number of samples, L, to approximate the unobserved predictions by $p(\hat{r}^d_{m,n',k'}|\mathcal{R}^d) \approx \sum_{l=1}^{L} p(\hat{r}^d_{m,n',k'}|\mathbf{u}^d_m[l], \mathbf{v}^d_{n'}[l], \mathbf{t}^d_{k'}[l], \alpha[l])$. To implement above described approaches, SRTF improves MCMC of BPTF, and it works as follows (please also see the Appendix section that describes our MCMC procedure in detail):

(1) Initializes $\mathbf{U}^d[1]$ and $\mathbf{V}^d[1]$ by MAP results of PMF as per BPTF. It also initializes $\mathbf{T}^d[1]$, $\mathbf{C}^v[1]$, and $\mathbf{C}^t[1]$ by Gaussian distribution. $\mathbf{X}$ is matrix representation of feature vector $\mathbf{x}$ (e.g. $\mathbf{C}^v[1]$ is the matrix representation of $\mathbf{c}^v_j$). $\mathbf{C}^v$ and $\mathbf{C}^t$ are necessary for implementing our approach (A). Next, it repeats steps (2) to (6) L times.

(2) Samples the hyperparameters the same way as BPTF i.e.:
- $\alpha[l] \sim p(\alpha[l]|\mathbf{U}^d[l], \mathbf{V}^d[l], \mathbf{T}^d[l], \mathcal{R}^d)$
- $\Theta_{U^d}[l] \sim p(\Theta_{U^d}[l]|\mathbf{U}^d[l])$
- $\Theta_{V^d}[l] \sim p(\Theta_{V^d}[l]|\mathbf{V}^d[l])$
- $\Theta_{T^d}[l] \sim p(\Theta_{T^d}[l]|\mathbf{T}^d[l])$
- $\Theta_{C^v}[l] \sim p(\Theta_{C^v}[l]|\mathbf{C}^v[l])$
- $\Theta_{C^t}[l] \sim p(\Theta_{C^t}[l]|\mathbf{C}^t[l])$

here, $\Theta_{\mathbf{X}}$ represents $\{\mu_{\mathbf{X}}, \Lambda_{\mathbf{X}}\}$. $\mu_{\mathbf{X}}$ and $\Lambda_{\mathbf{X}}$ are computed the same way as BPTF (see Preliminary).

(3) Samples the feature vectors the same way as BPTF as:
- $\mathbf{u}^d_m[l{+}1] \sim p(\mathbf{u}^d_m|\mathbf{V}^d[l], \mathbf{T}^d[l], \alpha[l], \Theta_{U^d}[l], \mathcal{R}^d)$
- $\mathbf{v}^d_{n'}[l{+}1] \sim p(\mathbf{v}^d_{n'}|\mathbf{U}^d[l{+}1], \mathbf{T}^d[l], \alpha[l], \Theta_{V^d}[l], \mathcal{R}^d)$
- $\mathbf{t}^d_{k'}[l{+}1] \sim p(\mathbf{t}^d_{k'}|\mathbf{U}^d[l{+}1], \mathbf{V}^d[l{+}1], \alpha[l], \Theta_{T^d}[l], \mathcal{R}^d)$

(4) Samples the semantically-biased feature vectors by using $\alpha[l]$, $\mathbf{U}^d[l{+}1]$, $\mathbf{V}^d[l{+}1]$, and $\mathbf{T}^d[l{+}1]$ as follows:
- $\mathbf{c}^v_j[l{+}1] \sim p(\mathbf{c}^v_j|\mathbf{U}^d[l{+}1], \mathbf{T}^d[l{+}1], \alpha[l], \Theta_{C^v}[l], \mathcal{R}^v)$
- $\mathbf{c}^t_j[l{+}1] \sim p(\mathbf{c}^t_j|\mathbf{U}^d[l{+}1], \mathbf{V}^d[l{+}1], \alpha[l], \Theta_{C^t}[l], \mathcal{R}^t)$

Parameters $\alpha[l]$, $\mathbf{U}^d[l{+}1]$, $\mathbf{V}^d[l{+}1]$, and $\mathbf{T}^d[l{+}1]$ are shared by step (3) and (4) to satisfy our approach (B).

(5) Samples the unobserved ratings $\hat{r}^d_{m,n',k'}[l]$ by applying $\mathbf{u}^d_m[l{+}1]$, $\mathbf{v}^d_{n'}[l{+}1]$, and $\mathbf{t}^d_{k'}[l{+}1]$ to Eq. (1).

(6) Updates $\mathbf{v}^d_{n'}[l{+}1]$ to $\mathbf{v}'_{n'}[l{+}1]$ by replacing $\mathbf{v}^d_{n'}[l{+}1]$ with $\mathbf{c}^v_{n'}[l{+}1]$ if $v_{n'}$ is a sparse item instance. In the same way, it updates $\mathbf{t}^d_{k'}[l{+}1]$ to $\mathbf{t}'_{k'}[l{+}1]$ if $t_{k'}$ is a sparse tag instance. This step is necessary for implementing our approach (C). SRTF regards $\mathbf{v}'_{n'}[l{+}1]$ as $\mathbf{v}^d_{n'}[l{+}1]$ and $\mathbf{t}'_{k'}[l{+}1]$ as $\mathbf{t}^d_{k'}[l{+}1]$ for the next iteration.

It finally computes the prediction for unobserved rating in original tensor $\hat{r}_{m,n,k}$ from $\hat{r}^d_{m,n',k'}$ by checking the link relationship between v_n and $v_{n'}$ and that between t_k and $t_{k'}$. Basically, our MCMC procedure only adds steps (4) and (6) to the original MCMC procedure of BPTF, but it does require several additional parameters such as $\mathbf{C}^v$ and $\mathbf{C}^t$. Thus readers can easily implement our method to advance their own research.

The complexity of SRTF in each MCMC iteration is $O(\#nz \times D^2 + (M + N' + K' + S^V + S^T) \times D^3)$. Because the first term is much larger than the rest, the computation time is almost the same as that of BPTF. The parameter δ and parameters for factorizations can be easily set based on the long-tail characteristic and the full Bayesian treatment inherited by the BPTF framework, respectively.

Table 1: Tag class examples for MovieLens.

C	Breathtaking	Historical	Spirited	Unrealistic
t_k	Breathtaking	Ancient	Racy	Kafkaesque
	Exciting	Historic	Spirited	Surreal
	Thrilling	Past life	Vibrant	Surreal life

Table 2: Mean/variance(x10^{-5}) of RMSEs for MovieLens.

		BPMF	*BPTF*	*SRTF*α	*SRTF*β	*SRTF(1.0)*	*SRTF(0.3)*
D=25	Mean	0.9062	1.0278	0.8842	0.8838	0.8850	0.8807
	Variance	8.1712	15.3502	4.7174	5.7223	5.4754	5.9744
D=50	Mean	0.9043	0.9988	0.8827	0.8818	0.8837	0.8791
	Variance	9.1206	39.4048	5.9430	4.6317	5.3072	6.8149
D=75	Mean	0.9040	0.9863	0.8821	0.8809	0.8829	0.8778
	Variance	6.9771	37.2929	5.0868	5.7490	5.5913	6.4982
D=100	Mean	0.9032	0.9708	0.8823	0.8792	0.8824	***0.8774***
	Variance	7.5844	14.5400	5.3995	5.3136	6.0363	6.1852

Evaluation

We applied SRTF to predict users' rating activities.

Dataset

The method's accuracy was confirmed using two datasets:
MovieLens contains ratings by users against movie items with user-assigned tags. Ratings range from 0.5 to 5. We created composite classes from the vocabulary of genres, that of directors, and that of actors gathered from Freebase and a tag taxonomy from WordNet. We restrict each item instance to not more than one director and two actors. As a result, item vocabulary has 724 composite classes. The taxonomy has 4,284 classes. We also used items and tags that do not have any classes. Consequently, it contains 24,565 ratings with 44,595 tag assignments; 33,547 tags have tag classes. The size of user-item-tag tensor is $2,026 \times 5,088 \times 9,160$. Table 1 shows examples of class C and its tag instances $t_{k'}$s.
Yelp contains ratings assigned by users to restaurants in their reviews. Ratings range from 1 to 5. We used the genre vocabulary of Yelp as the item vocabulary. It has 179 classes. We do not create composite classes because it offers only one useful property type, genre. We analyzed review texts by using the food vocabulary in DBPedia[3] as (Nakatsuji, Yoshida, and Ishida 2009) did, so this dataset does not require disambiguation. We extracted 2,038,560 food phrases. The food phrase vocabulary contains 1,957 distinct phrases in 2,316 classes. it contains 158,424 ratings with reviews. Among those, 33,863 entries do not contain the phrases, thus we assign dummy phrase id for those. By doing so, we can use all restaurant reviews. The size of user-item-tag tensor is $36,472 \times 4,503 \times 1,957$.

Compared methods Compared methods are: (1) *BPMF*; the Bayesian probabilistic matrix factorization that analyzes ratings by users for items without tags (Salakhutdinov and Mnih 2008a); it can not predict ratings with tags though they assist users in understanding the items (Rendle et al. 2009). (2) *BPTF* (Xiong et al. 2010). (3) *SRTF*α; it randomly links an ambiguous object to one of its instances whose names are same with that of object. (4) *SRTF*β; does not use composite classes. (5) *SRTF(1.0)*; sets δ to 1.0 and abstracts all objects to their classes. (6) *SRTF(0.3)*; sets δ to 0.3 following the long-tail characteristics. (7) *GCTF*; the most popular method for factorizing several tensors/matrices simultaneously (Ermis, Acar, and Cemgil 2012).

[3]http://dbpedia.org/ontology/Food

Table 3: Mean/variance(x10^{-4}) of RMSEs for Yelp.

		BPMF	*BPTF*	*SRTF*α	*SRTF*β	*SRTF(1.0)*	*SRTF(0.3)*
D=25	Mean	1.1161	1.2222	-	-	1.2240	1.0858
	Variance	9.5985	1.0143	-	-	1.5127	1.4318
D=50	Mean	1.1154	1.1643	-	-	1.1613	1.0860
	Variance	10.1788	0.9400	-	-	1.4273	1.2976
D=75	Mean	1.1147	1.1620	-	-	1.1627	1.0856
	Variance)	10.4442	1.2118	-	-	1.6251	1.3544
D=100	Mean	1.1127	1.1533	-	-	1.1621	***1.0851***
	Variance	10.0000	0.2688	-	-	1.8292	1.3444

Table 4: Comparing our method with *GCTF* ($D = 50$).

MovieLens			Yelp		
BPTF	*SRTF(0.3)*	*GCTF*	*BPTF*	*SRTF(0.3)*	*GCTF*
0.9243	0.8791	0.9947	1.7821	1.1903	1.4192

Methodology and parameter setup Following the methodology used in the BPTF paper (Xiong et al. 2010), we used Root Mean Square Error (RMSE), computed by $\sqrt{(\sum_{i=1}^{n}(P_i - R_i)^2)/n}$, where n is the number of entries in the test dataset. P_i and R_i are the predicted and actual ratings of the i-th entry, respectively. Smaller RMSE values indicate higher accuracy. We divided the dataset into three parts and performed three-fold cross validation. Results below are averages with variance values of the three evaluations. Following (Xiong et al. 2010), the parameters are μ_0=0, ν_0=0, β_0=0, $\mathbf{W}_0$=$\mathbf{I}$, $\tilde{\nu}_0$=1, and $\tilde{W}_0$=1. L is 500. We used the IS cost function for *GCTF* because it achieved the highest accuracy.

Results We first compared the accuracy of the methods (mean and variance values for three-fold cross validation) as shown in Table 2 and 3. *BPTF* has worse accuracy than *BPMF*. This indicates that *BPTF* can not utilize additional information (tags/reviews) for predictions because observations in tensors become much sparser than those of matrices. Interestingly, *SRTF*α has much better accuracy (lower RMSE) than *BPTF* even though it uses a simple disambiguation strategy. This is because many item objects are not ambiguous and the improvement by abstraction from non-ambiguous objects greatly exceeds the decrease created by disambiguation mistakes in *SRTF*α. We note that this result showed that the abstraction idea is useful even with a simple disambiguation process. This means that our method will positively impact many applications. *SRTF*β has worse accuracy than *SRTF(0.3)* because *SRTF*β does not have composite classes. It does not use actor or director properties in item vocabularies. Furthermore, *SRTF(1.0)* has worse accuracy than *SRTF(0.3)*. This indicates that it is not useful to incorporate semantics into non-sparse objects. *SRTF(1.0)* in Yelp has poor accuracy because its item/tag vocabularies are less detailed than the vocabulary/taxonomy of MovieLens. Finally, *SRTF(0.3)*, which provides sophisticated disambiguation/augmentation strategies, improved the accuracy more than the other methods with the statistical significance of α<0.05. *SRTF(0.3)* achieves 10% higher accuracy than BPTF for MovieLens (*SRTF(0.3)*

Table 5: Prediction examples for MovieLens and Yelp by *BPTF*/*SRTF(0.3)* (bold words represent food tags).

Training dataset			Rating predictions by *BPTF* and *SRTF(0.3)*				
Tag in review sentence	Item/main genre	Rating	Tag in review sentence	Item/main genre	BPTF	SRTF	Actual
President	Air Force One/Thriller&Wolfgang Peterson	3.5	disease	Outbreak/Thriller&Wolfgang Peterson	2.5	3.4	3.5
History	All the Presidents Men/Historical Fiction	4.0	Historical	Glory/Historical Fiction	3.0	3.7	4.0
Tempura ***udon*** was delicious.	A/Japanese	4.0	***Ramen*** was really good.	B/Hawaiian	5.0	3.8	4.0
We enjoyed the ***foie gras***.	C/French	4.0	***Gratin*** is excellent.	D/Steakhouses	2.8	3.7	4.0

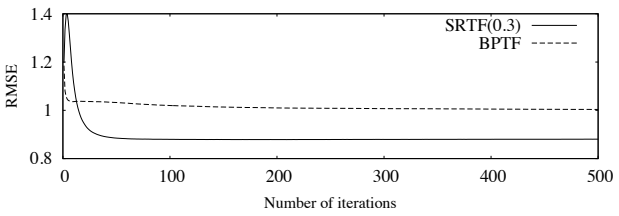

Figure 3: RMSE vs. number of iterations ($D = 50$).

marked 0.8774 while *BPTF* marked 0.9708.) We can also confirm that *SRTF(0.3)* output more stable predictions than *BPTF* or *BPMF* because variance values of *SRTF(0.3)* are much smaller than those of *BPTF* and *BPMF*. The variance values of *BPTF* is smaller than those of *SRTF(0.3)*, however, *SRTF(0.3)* output much accurate results than *BPTF*.).

We next compared the accuracy of *BPTF*, *SRTF(0.3)*, and *GCTF*. Since *GCTF* requires much more computation than *BPTF* or *SRTF* (see Preliminary), we could not apply *GCTF* to the whole evaluation dataset on our computer [4]. Accordingly, we randomly picked five sets of 200 users in each dataset and evaluated the accuracy for each set. Table 4 shows the average RMSE values of those evaluations. *SRTF* achieves higher accuracy than *GCTF*. This is because *SRTF* naturally incorporates semantic classes into tensors after fitting them to user-item-tag tensor; *GCTF* straightforwardly combines different types of multi-object relationships, rating relationships among users, items, and tags, link relationships among items and their classes, and link relationships among tags and their classes. *SRTF* also carefully selects sparse objects and gives semantic biases to them; *GCTF* gives biases from extra information to all multi-object relationships.

Fig. 3 presents the accuracy on MovieLens when the number of iterations was changed in our MCMC procedures. This confirms that the accuracy of *SRTF* converged quickly because it uses the MAP results from PMF to optimize the initializations of $\mathbf{U}$ and $\mathbf{V}$. Yelp result had similar tendencies.

We show the examples of the differences between the predictions output by *BPTF* and those by *SRTF(0.3)* in Table 5. The column "BPTF", "SRTF", and "Actual" present prediction vales by *BPTF*, those by *SRTF(0.3)*, and actual ratings given by users as found in the test dataset, respectively. MovieLens item "Outbreak" was rarely observed in the training dataset. *SRTF(0.3)* predicted this selection accurately since it can use the knowledge that the combination of thriller movies directed by "Wolfgang Petersen" and historical movies like "All the Presidents Men" tended to be highly rated in the training dataset. In Yelp dataset, the combination of tag "udon" at restaurant "A" and "foie gras" at restaurant "C" were highly rated in the training dataset. In the test dataset, the combination of tag "ramen" at restaurant "B" and "gratin" at restaurant "D" were highly rated. Those tags are sparsely observed in the training dataset. *SRTF(0.3)* accurately predicted those selections since it uses knowledge that tags "udon" and "ramen" are both in tag class "Japanese noodles", as well as the knowledge that tags "foie gras" and "gratin" are both in tag class "French dishes". Note that restaurant "B" is not a Japanese restaurant, and that restaurant "D" is not a French restaurant. Thus, those selections were accurately predicted via tag classes and not by item classes. *BPTF* predictions were inaccurate since they could not use the semantics behind the objects being rated.

Since our disambiguation process is pre-computed before tensor factorization, and our augmentation process is quite simple, they can be easily applied to other tensor factorization schemes. For example, we have confirmed that our approach works well when applied to another popular tensor factorization scheme, non-negative tensor factorization (NTF); we extended the variational non-negative matrix factorization in (Cemgil 2009) to variational non-negative tensor factorization and applied our ideas to NTF. (Cemgil 2009) introduced the NMF in a Poisson model that fits the exponential family distribution often seen in various kinds of datasets. Thus, our ideas can enhance the prediction accuracy for various kinds of applications.

Conclusions and directions for future work

This is the first study to apply the semantics behind objects to enhance tensor factorization accuracy. Semantic data representation for tensor factorization is critical in using semantics to analyze human activities; a key AI goal. It first creates semantically enhanced tensors and then factorizes the tensors simultaneously over the BPTF framework. Experiments showed that SRTF achieves 10% higher accuracy than current methods and can support many applications.

We are now applying SRTF to link predictions in social networks (e.g. predicting the frequency of future communications among users). Accordingly, we are applying our idea to NTF as described in the evaluation section because the frequency of communications among users often follows the exponential family distribution. Another interesting direction for our future work is predicting user activities among cross-domain applications such as music and movie rating services. We think our ideas have potential for cross domain

[4]Recently, (Ermis, Acar, and Cemgil 2013) improved the time complexity of GCTF by taking into account the sparsity pattern of the dataset.

analysis because they allow for the effective use of semantic knowledge in the format of LOD shared among several service domains. The third interesting direction is that the development of methods that handle more detailed semantic knowledge other than simple vocabularies/taxonomies.

References

Acar, E.; Kolda, T. G.; and Dunlavy, D. M. 2011. All-at-once optimization for coupled matrix and tensor factorizations. *CoRR* abs/1105.3422.

Anderson, C. 2006. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; and Hellmann, S. 2009. Dbpedia - a crystallization point for the web of data. *Web Semant.* 7(3):154–165.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.* 5(3):1–22.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. SIGMOD'08*, 1247–1250.

Cantador, I.; Konstas, I.; and Jose, J. M. 2011. Categorising social tags to improve folksonomy-based recommendations. *Web Semant.* 9(1):1–15.

Cemgil, A. T. 2009. Bayesian inference for nonnegative matrix factorisation models. *Intell. Neuroscience* 2009:4:1–4:17.

Ermis, B.; Acar, E.; and Cemgil, A. T. 2012. Link prediction via generalized coupled tensor factorisation. *CoRR* abs/1208.6231.

Ermis, B.; Acar, E.; and Cemgil, A. T. 2013. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining and Knowledge Discovery*.

Gabrilovich, E., and Markovitch, S. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. IJCAI'07*, 1606–1611.

Hu, J.; Fang, L.; Cao, Y.; Zeng, H.-J.; Li, H.; Yang, Q.; and Chen, Z. 2008. Enhancing text clustering by leveraging wikipedia semantics. In *Proc. SIGIR'08*, 179–186.

Jain, P.; Hitzler, P.; Yeh, P. Z.; Verma, K.; and Sheth, A. P. 2010. A.p.: Linked data is merely more data. In *Proc. AAAI Spring Symposium 'Linked Data Meets Artificial Intelligence'*, 82–86.

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

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *Proc. ACL'03*, 423–430.

McGuinness, D. L. 2003. Ontologies come of age. In *Spinning the Semantic Web*, 171–194. MIT Press.

Miller, G. A. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

Nakatsuji, M., and Fujiwara, Y. 2014. Linked taxonomies to capture users' subjective assessments of items to facilitate accurate collaborative filtering. *Artificial Intelligence* 207(0):52 – 68.

Nakatsuji, M.; Fujiwara, Y.; Uchiyama, T.; and Toda, H. 2012. Collaborative filtering by analyzing dynamic user interests modeled by taxonomy. In *Proc. ISWC'12*, 361–377.

Nakatsuji, M.; Yoshida, M.; and Ishida, T. 2009. Detecting innovative topics based on user-interest ontology. *Web Semant.* 7(2):107–120.

Nakaya, A.; Katayama, T.; Itoh, M.; Hiranuka, K.; Kawashima, S.; Moriya, Y.; Okuda, S.; Tanaka, M.; Tokimatsu, T.; Yamanishi, Y.; Yoshizawa, A. C.; Kanehisa, M.; and Goto, S. 2013. Kegg oc: a large-scale automatic construction of taxonomy-based ortholog clusters. *Nucleic Acids Research* 41(Database-Issue):353–357.

Narita, A.; Hayashi, K.; Tomioka, R.; and Kashima, H. 2011. Tensor factorization using auxiliary information. In *Proc. ECML-PKDD'11*, 501–516.

Parundekar, R.; Knoblock, C. A.; and Ambite, J. L. 2012. Discovering concept coverings in ontologies of linked data sources. In *Proc. ISWC'12*, 427–443.

Rendle, S., and Schmidt-Thieme, L. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proc. WSDM'10*, 81–90.

Rendle, S.; Marinho, L. B.; Nanopoulos, A.; and Schmidt-Thieme, L. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *Proc. KDD'09*, 727–736.

Saif, H.; He, Y.; and Alani, H. 2012. Semantic sentiment analysis of twitter. In *Proc. ISWC'12*, 508–524.

Salakhutdinov, R., and Mnih, A. 2008a. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proc. ICML'08*, volume 25.

Salakhutdinov, R., and Mnih, A. 2008b. Probabilistic matrix factorization. In *Proc. NIPS'08*, volume 20.

Takeuchi, K.; Tomioka, R.; Ishiguro, K.; Kimura, A.; and Sawada, H. 2013. Non-negative multiple tensor factorization. In *Proc. ICDM'13*, 1199–1204.

Turney, P. D., and Pantel, P. 2010. From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.* 37(1):141–188.

Xiong, L.; Chen, X.; Huang, T.-K.; Schneider, J. G.; and Carbonell, J. G. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proc. SDM'10*, 211–222.

Yilmaz, Y. K.; Cemgil, A.-T.; and Simsekli, U. 2011. Generalised coupled tensor factorisation. In *Proc. NIPS'11*, 2151–2159.

Yu, J.; Zha, Z.-J.; Wang, M.; and Chua, T.-S. 2011. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proc HLT '11*, 1496–1505.

Zheng, V. W.; Cao, B.; Zheng, Y.; Xie, X.; and Yang, Q. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proc. AAAI'10*, 236–241.

Zheng, J. G.; Fu, L.; Ma, X.; and Fox, P. 2013. Sem+: Discover "same as" links among the entities on the web of data. Technical report, Rensselaer Polytechnic Institute, http://tw.rpi.edu/web/doc/Document?uri=-http://tw.rpi.edu/media/2013/10/07/e293/SEM.doc .

Appendix

This section explains how to learn feature vectors and their hyperparameters in the MCMC procedure of SRTF in detail. Please also refer to section "Implementation over the BPTF framework". The following notations and item numbers are the same as those used in the section "Implementation over the BPTF framework". The MCMC procedure of SRTF improves BPTF following the approaches explained in the section "Approach" in the paper. A detailed explanation of MCMC of BPTF is given in the BPTF paper (Xiong et al. 2010).

SRTF applies $\mathbf{u}^d_m$, $\mathbf{v}'_{n'}$, and $\mathbf{t}'_{k'}$, and their hyperparameters to Eq. (2) in the paper. It learns the predictive distribution for unobserved ratings $\hat{r}^d_{m,n',k'}$ by MCMC procedure as per BPTF. Especially, MCMC collects a number of samples, L, to approximate the unobserved predictions for $\mathcal{R}^d$ by $p(\hat{r}^d_{m,n',k'}|\mathcal{R}^d) \approx \sum_{l=1}^{L} p(\hat{r}^d_{m,n',k'}|\mathbf{u}^d_m[l], \mathbf{v}^d_{n'}[l], \mathbf{t}^d_{k'}[l], \alpha[l])$. The MCMC procedure of SRTF works as follows:

(1) It first initializes $\mathbf{U}^d[1]$, $\mathbf{V}^d[1]$ by MAP results of PMF as per BPTF. It also initializes $\mathbf{T}^d[1]$, $\mathbf{C}^v[1]$, and $\mathbf{C}^t[1]$ by Gaussian distribution. $\mathbf{C}^v$ and $\mathbf{C}^t$ are matrix representations of $\mathbf{c}^v_j$ and $\mathbf{c}^t_j$, respectively. Next, it repeats steps (2) to (6) L times and samples feature vectors and their hyperparameters. Each sample in MCMC procedure depends only on the previous one.

(2) It samples the conditional distribution of α given $\mathcal{R}^d$, $\mathbf{U}^d$, $\mathbf{V}^d$, and $\mathbf{T}^d$ as per BPTF as follows:

$$p(\alpha|\mathcal{R}^d, \mathbf{U}^d, \mathbf{V}^d, \mathbf{T}^d) = \mathcal{W}(\alpha|\tilde{W}^*_0, \nu^*_0),$$
$$\nu^*_0 = \tilde{\nu}_0 + \sum_{k'=1}^{K'}\sum_{n'=1}^{N'}\sum_{m=1}^{M} 1, (\tilde{W}^*_0) = (\tilde{W}_0)^{-1}$$
$$+ \sum_{k'=1}^{K'}\sum_{n'=1}^{N'}\sum_{m=1}^{M}(r^d_{m,n',k'} - \langle \mathbf{u}^d_m, \mathbf{v}^d_{n'}, \mathbf{t}^d_{k'}\rangle)^2.$$

It also samples hyperparameters Θ_{U^d}, Θ_{V^d}, and Θ_{T^d} as per BPTF. They are computed in the same way. For example, Θ_{V^d} is conditionally independent of all other parameters given $\mathbf{V}^d$; it is computed as follows:

$$p(\Theta_{V^d}|\mathbf{V}^d) = \mathcal{N}(\mu_{V^d}|\mu^*_0, (\beta^*_0\Lambda_{V^d})^{-1})\mathcal{W}(\Lambda_{V^d}|\mathbf{W}^*_0, \nu^*_0),$$
$$\mu^*_0 = \tfrac{\beta_0\mu_0 + N'\bar{\mathbf{v}}^d}{\beta_0+N'}, \beta^*_0 = \beta_0 + N', \nu^*_0 = \nu_0 + N',$$
$$\mathbf{W}^{*-1}_0 = \mathbf{W}_0{}^{-1} + N'\bar{\mathbf{S}} + \tfrac{\beta_0 N'}{\beta_0+N'}(\nu_0 - \bar{\mathbf{v}}^d)(\nu_0 - \bar{\mathbf{v}}^d)^{tr},$$
$$\bar{\mathbf{v}}^d = \tfrac{1}{N'}\sum_{n'=1}^{N'} \mathbf{v}^d_{n'},$$
$$\bar{\mathbf{S}} = \tfrac{1}{N'}\sum_{n'=1}^{N'} (\mathbf{v}^d_{n'} - \bar{\mathbf{v}}^d)(\mathbf{v}^d_{n'} - \bar{\mathbf{v}}^d)^{tr}.$$

Our MCMC procedure also samples the hyperparameter of $\mathbf{c}^v$, Θ_{C^v}, to generate ratings for $\mathcal{R}^v$. The generative process of $\mathcal{R}^v$ is almost the same as that of $\mathcal{R}^d$ except for sharing α, $\mathbf{u}^d_m$, and $\mathbf{t}^d_{k'}$, which are also used in sampling $\mathbf{v}^d_{n'}$, in sampling $\mathbf{c}^v_j$ (see steps (3) and (4)). Thus, Θ_{C^v} can be computed in the same way as Θ_{V^d} as follows:

$$p(\Theta_{C^v}|\mathbf{C}^v) = \mathcal{N}(\mu_{C^v}|\mu^*_0, (\beta^*_0\Lambda_{C^v})^{-1})\mathcal{W}(\Lambda_{C^v}|\mathbf{W}^*_0, \nu^*_0),$$
$$\mu^*_0 = \tfrac{\beta_0\mu_0 + (N'+S^V)\bar{\mathbf{c}^v}}{\beta_0+N'+S^V}, \beta^*_0 = \beta_0 + N' + S^V, \nu^*_0 = \nu_0 + N' + S^V,$$
$$\mathbf{W}^{*-1}_0 = \mathbf{W}_0{}^{-1} + (N'+S^V)\bar{\mathbf{S}} + \tfrac{\beta_0(N'+S^V)}{\beta_0+N'+S^V}(\nu_0 - \bar{\mathbf{c}^v})(\nu_0 - \bar{\mathbf{c}^v})^{tr},$$
$$\bar{\mathbf{c}^v} = \tfrac{1}{N'+S^V}\sum_{j=1}^{N'+S^V} \mathbf{c}^v_j,$$
$$\bar{\mathbf{S}} = \tfrac{1}{N'+S^V}\sum_{j=1}^{N'+S^V} (\mathbf{c}^v_j - \bar{\mathbf{c}^v})(\mathbf{c}^v_j - \bar{\mathbf{c}^v})^{tr}.$$

(3) Then it samples model parameters $\mathbf{U}^d$, $\mathbf{V}^d$, $\mathbf{T}^d$ as per BPTF. They are computed in the same way. For example, $\mathbf{V}^d$ can be factorized into individual items. If $\mathbf{P}_{m,k'} \equiv \mathbf{u}^d_m \cdot \mathbf{t}^d_{k'}$, each item feature vector is computed in parallel as follows:

$$p(\mathbf{v}^d_{n'}|\mathcal{R}^d, \mathbf{U}^d, \mathbf{T}^d, \alpha, \Theta_{V^d}) = \mathcal{N}(\mathbf{v}^d_{n'}|\mu^*_{n'}, (\Lambda^*_{n'})^{-1}),$$
$$\mu^*_{n'} \equiv (\Lambda^*_{n'})^{-1}(\Lambda_{V^d}\mu_{V^d} + \alpha\sum_{k'=1}^{K'}\sum_{m=1}^{M} r^d_{m,n',k'}\mathbf{P}_{m,k'}),$$
$$\Lambda^*_{n'} = \Lambda_{V^d} + \alpha\sum_{k'=1}^{K'}\sum_{m=1}^{M} o^d_{m,n',k'}\mathbf{P}_{m,k'}\mathbf{P}^{tr}_{m,k'}.$$

(4) It also samples model parameters $\mathbf{C}^v$ and $\mathbf{C}^t$. Conditional distribution of $\mathbf{C}^v$ can also be factorized into individual items and individual augmented item classes as follows:

$$p(\mathbf{c}^v_j|\mathcal{R}^v, \mathbf{U}^d, \mathbf{T}^d, \alpha, \Theta_{C^v}) = \mathcal{N}(\mathbf{c}^v_j|\mu^*_j, (\Lambda^*_j)^{-1}),$$
$$\mu^*_j \equiv (\Lambda^*_j)^{-1}(\Lambda_{C^v}\mu_{C^v} + \alpha\sum_{k'=1}^{K'}\sum_{m=1}^{M} r^v{}_{m,j,k'}\mathbf{P}_{m,k'}),$$
$$\Lambda^*_j = \Lambda_{C^v} + \alpha\sum_{k'=1}^{K'}\sum_{m=1}^{M} o^v_{m,j,k'}\mathbf{P}_{m,k'}\mathbf{P}^{tr}_{m,k'}.$$

Here, $o^v_{m,j,k'}$ is a 0/1 flag to indicate the observation existence of multi-object relationships composed by user u_m, j-th item/class that corresponds to $\mathbf{c}^v_j$, and tag $t_{k'}$ in $\mathcal{R}^v$.

Steps (3) and (4) share precision α and feature vectors $\mathbf{u}^d_m$ and $\mathbf{t}^d_{k'}$ (in $\mathbf{P}_{m,k'}$). This means SRTF shares those parameters in the factorization of tensors $\mathcal{R}^d$ and $\mathcal{R}^v$. Thus, semantic biases can be shared among the above tensors via those shared parameters. The conditional distribution of $\mathbf{C}^t$ can also be computed in the same way.

Steps (3) and (4) also indicate that feature vectors for objects as well as semantically-biased feature vectors can be computed in parallel.

(5) In each iteration, it samples the unobserved ratings $\hat{r}^d_{m,n',k'}$ by applying $\mathbf{u}^d_m$, $\mathbf{v}^d_{n'}$, and $\mathbf{t}^d_{k'}$ to Eq. (1) in the paper.

(6) It updates $\mathbf{v}^d_{n'}$ to $\mathbf{v}'_{n'}$ by replacing $\mathbf{v}^d_{n'}$ with $\mathbf{c}^v_{n'}$ if $v_{n'}$ is a sparse item instance. Otherwise, it does not replace $\mathbf{v}^d_{n'}$ with $\mathbf{c}^v_{n'}$. In the same way, it updates $\mathbf{t}^d_{k'}$ to $\mathbf{t}'_{k'}$ if $t_{k'}$ is a sparse tag instance. SRTF regards $\mathbf{v}'_{n'}$ as $\mathbf{v}^d_{n'}$ and $\mathbf{t}'_{k'}$ as $\mathbf{t}^d_{k'}$ for the next iteration.

SRTF finally computes the prediction value for unobserved rating in original tensor $\hat{r}_{m,n,k}$ from $\hat{r}^d_{m,n',k'}$ by checking link relationship between v_n and $v_{n'}$ and that between t_k and $t_{k'}$.

In this way, SRTF effectively incorporates semantic biases into the feature vectors of sparse objects in each iteration of its MCMC procedure. It greatly solves the sparsity problem.