Semantic Sensitive Simultaneous Tensor Factorization

Makoto Nakatsuji

NTT Resonant Inc.,
Granparktower, 3-4-1 Shibaura, Minato-ku, Tokyo 108-0023, Japan,
nakatuji@nttr.co.jp

Abstract. The semantics distributed over large-scale knowledge bases can be used to intermediate heterogeneous users' activity logs created in services; such information can be used to improve applications that can help users to decide the next activities/services. Since user activities can be represented in terms of relationships involving three or more things (e.g. a user tags movie items on a webpage), tensors are an attractive approach to represent them. The recently introduced Semantic Sensitive Tensor Factorization (SSTF) is promising as it achieves high accuracy in predicting users' activities by basing tensor factorization on the semantics behind objects (e.g. item categories). However, SSTF currently focuses on the factorization of a tensor for a single service and thus has two problems: (1) *the balance problem* occurs when handling heterogeneous datasets simultaneously, and (2) *the sparsity problem* triggered by insufficient observations within a single service. Our solution, Semantic Sensitive Simultaneous Tensor Factorization (S^3TF), tackles the problems by: (1) Creating tensors for individual services and factorizing them simultaneously; it does not force the creation of a tensor from multiple services and factorize the single tensor. This avoids the low prediction accuracy caused by the balance problem. (2) Utilizing shared semantics behind distributed activity logs and assigning semantic bias to each tensor factorization. This avoids the sparsity problem by sharing semantics among services. Experiments using real-world datasets show that S^3TF achieves higher accuracy in rating prediction than the current best tensor method. It also extracts implicit relationships across services in the feature spaces by simultaneous factorization with shared semantics.

1 Introduction

Recently, many large-scale knowledge bases (KBs) have been constructed, including academic projects such as YAGO [8], DBpedia [2], and Elementary/ Deep-Dive [15], and commercial projects, such as those by Google [6] and Walmart [4]. These knowledge repositories hold millions of facts about the world, such as information about people, places, and things. Such information is deemed essential for improving AI applications that require machines to recognize and understand queries and their semantics in search or question answering systems. The applications include Google search and IBM's Watson, as well as smart

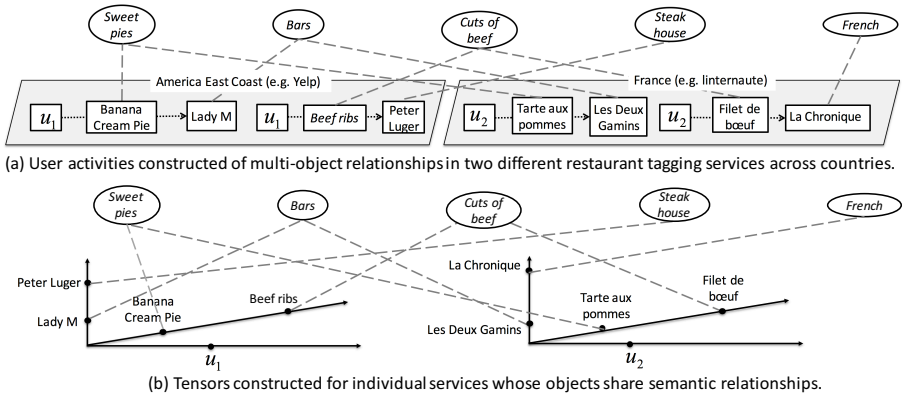


Fig. 1. Creating tensors for individual services whose objects are linked by semantics.

mobile assistants such as Apple's Siri and NTT docomo's Shabette-Concier [5]. They now assist users to acquire meaningful knowledge in their daily activities; e.g. looking up an actor's birthday by question-answering systems or searching restaurants near the user's current location by smart mobile assistants.

The KBs can also be used to provide background knowledge that is shared by the different services [2]. Thus, beyond the above described usages of facts stored in the KBs, the semantics in those bases can be effectively used for mediating distributed users' activity logs in different services. Thus they have the potential to let AI applications assist users to decide next activities across services by analyzing heterogeneous users' logs distributed across services. In this paper, we assume services are different with each other if they do not share any objects, e.g. users, venues, or reviews. For example, in Fig. 1, US restaurant review service, Yelp[1], and French one, linternaute[2], are quite different services.

Tensor factorization methods have become popular for analyzing users' activities, since users' activities can be represented in terms of relationships involving three or more things (e.g. when a user tags venues on a webpage) [9,11,13,17,20]. Among the proposals made to date, Bayesian Probabilistic Tensor Factorization (BPTF) [20] is promising because of its efficient sampling of large-scale datasets and simple parameter settings. Semantic Sensitive Tensor Factorization (SSTF) [11,13] extends BPTF and applies semantic knowledge in the form of vocabularies/taxonomies extracted from Linked Open Data (LOD) to tensor factorization to solve the sparsity problem caused by sparse observation of objects. By incorporating the semantics behind objects, SSTF achieves the best rating prediction accuracy among the existing tensor factorization methods [13].

However, SSTF can not enhance prediction accuracy across services for two reasons: (1) SSTF suffers from the *balance problem* that arises when handling heterogeneous datasets, e.g. the predictions for the smaller services are greatly biased by the predictions for the larger services [10]. Even if we merge user

[1] http://www.yelp.com
[2] http://www.linternaute.com/restaurant/

activity logs across services based on objects that appear across services, SSTF prediction results are poor when faced with merged logs. (2) SSTF focuses on only the factorization of a tensor representing users' activities within a single service and cannot solve *the sparsity problem*. Even if the logs in different services share some *semantic* relationships, SSTF can not make use of them.

We think that a tensor factorization method that uses the semantics in the KBs to intermediate different services is needed since LOD project aims to mediate distributed data in different services [1]. Thus, this is an important goal for the Semantic Web community. For example, we can simultaneously analyze logs in an American restaurant review service and those in an equivalent Japan service by using semantics even if they share no users, restaurant venues, and review descriptions. As a result, we can improve the prediction accuracy of the individual services, extract the implicit relationships across services, and recommend good Japanese restaurants to users in the United States (and vice verse). So, this paper enhances SSTF and proposes Semantic Sensitive Simultaneous Tensor Factorization (S^3TF) that simultaneously factorizes tensors created for different services by relying on the semantics shared among services. It overcomes the above mentioned problems by taking the following two ideas:

(1) It creates tensors for individual services whose objects are linked by semantics. This means that S^3TF does not force a tensor to be created from multiple services and then factorize that single tensor to make predictions. Below, for ease of understanding, this paper uses the scenario in which there are two different restaurant review services in different countries, (they share no users, restaurants, or food reviews); e.g. Yelp and linternaute in Fig. 1. Fig. 1-(a) presents an example of users' activities involving three objects: a user who assigned tags about impressive foods served by restaurants with ratings on those relationships. The restaurants and foods are linked by the semantics from the KB. In the figure, say American user u_1 assigned tag "Banana cream pie" to restaurant "Lady M". French user u_2 assigned tag "Tarte aux pommes" to restaurant "Les Deux Gamins". In Fig. 1-(b), S^3TF creates tensors for two different services while sharing semantic classes; e.g. Food "Banana cream pie" is linked with food class "Sweet pies" and restaurant "Lady M" is linked with restaurant class "Bars" in a tensor for "America East Coast". Food "Tarte aux pommes" is linked with the food class "Sweet pies" and restaurant "Les Deux Gamins" is linked with the restaurant class "Bars" in a tensor for "French". As a result, S^3TF can factorize those individual tensors "individually" while sharing semantics across tensors. This solves the *balance problem*.

(2) It uses the shared semantics present in distributed services and uses the semantics to bias the latent features learned in each service's tensor factorization. Thus, it can avoid *the sparsity problem* of tensor factorization, by using not only the semantics shared within a service but also those shared among services. This has another effect: the semantic biases are shared in latent features for the tensors of individual services and thus S^3TF can extract the implicit relationships among services present in the latent features. For example, in Fig. 1-(a), user u_1 and u_2 share no foods and no restaurants with each other, though they may share almost

the same tendencies in food choice (e.g. they both tend to eat "Sweet pies" at "Bars" and "Cuts of beef" at nice restaurants). If such multi-object relationships are sparsely observed in each country, they can not be well predicted by current tensor factorization methods because of the sparsity problem. S^{3}TF solves this by using the shared semantics among services. It propagates observations for "Banana cream pie" and "Tarte aux pommes" to the class "Sweet pie" as well as the observations for "Lady M" and "Les Deux Gamins" to the class "Bar". It then applies the semantic biases from food class "Sweet pie" to "Banana cream pie" as well as those from restaurant class "Bars" to restaurant "Lady M" when the tensor for United States is factorized. It also applies semantic biases from food class "Sweet pie" to "Tarte aux pommes" as well as those from restaurant class "Bars" to restaurant "Les Deux Gamins" when the tensor for France is factorized. In this way, S^{3}TF solves the sparsity problem by using the semantics shared across services. It also can find the implicit relationships from the latent features (e.g. the relationships shared by users u_1 and u_2 described above) by the mediation provided by the shared semantics.

We evaluated S^{3}TF using restaurant review datasets across countries. The reviews do not share any users, restaurant venues, or review descriptions as the languages are different. Thus, they are considered to be different services. The results show that S^{3}TF outperforms the previous methods including SSTF by sharing the semantics behind venues and review descriptions across services.

The paper is organized as follows: Section 2 describes related works while Section 3 introduces the background of this paper. Section 4 explains our method and Section 5 evaluates it. Finally, Section 6 concludes the paper.

2 Related work

Tensor factorization methods have recently been used in various applications such as recommendation systems [11, 17] and LOD analyses [7, 14]. For example, [14] proposed methods that use tensor factorization to analyze huge volumes of LOD datasets in a reasonable amount of time. They, however, did not use the simultaneous tensor factorization approach and thus could not explicitly incorporate the semantic relationships behind multi-object relationships into the tensor factorization; in particular, they failed to use taxonomical relationships behind multi-object relationships such as "subClassOf" and "subGenreOf", which are often seen in LOD datasets. A recent proposal, SSTF [11, 13], solves the sparsity problem by providing semantic bias from KBs to the feature vectors for sparse objects in multi-object relationships. SSTF was, however, not designed to perform cross-domain analysis even though LOD can be effectively used for mediating distributed objects in different services [2]. Generalized Coupled Tensor Factorization (GCTF) methods [22] and recent Non-negative Multiple Tensor Factorization (NMTF) [19] try to incorporate extra information into tensor factorization by simultaneously factorizing observed tensors and matrices representing extra information. They, however, do not focus on handling semantics behind objects while factorizing tensors created for different services. Further-

more, according to the evaluations in [13], they have much worse performance than SSTF.

Other than tensor methods, [23] applies embedding models including heterogeneous network embedding and deep learning embedding to automatically extract semantic representations from the KB. Then it jointly learns the latent representations in collaborative filtering as well as items' semantic representations from the KB. There are, however, no embedding methods that analyze different services by using shared KBs.

Recent semantic web studies try to find missing links between entities [21] or find an explanation on a pair of entities in KBs [16]. [12,18] incorporate semantic categories of items into the model and improve the recommendation accuracies. They, however, do not focus on the analysis of users' activities across services and find implicit relationships between entities by the above mentioned analysis.

3 Preliminary

Here, we explain Bayesian Probabilistic Tensor Factorization (BPTF) since S^{3}TF was implemented within the BPTF framework due to its efficiency with simple parameter settings.

This paper deals with the relationships formed by user u_m, venue v_n, and tag t_k. A third-order tensor $\mathcal{R}$ is used to model the relationships among objects from sets of users, venues, and tags. Here, the (m, n, k)-th element $r_{m,n,k}$ indicates the m-th user's rating of the n-th venue with the k-th tag. Tensor factorization assigns a D-dimensional latent feature vector to each user, venue, and tag, denoted as $\mathbf{u}_m$, $\mathbf{v}_n$, and $\mathbf{t}_k$, respectively. Here, $\mathbf{u}_m$ is an M-length, $\mathbf{v}_n$ is an N-length, and $\mathbf{t}_k$ is a K-length "column" vector. Accordingly, each element $r_{m,n,k}$ in $\mathcal{R}$ can be approximated as the inner-product of the three vectors as follows:

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

where index d represents the d-th "row" element of each vector.

BPTF [20] models tensor factorization over a generative probabilistic model for ratings with Gaussian/Wishart priors over parameters. The Wishart distribution is most commonly used as the conjugate prior for the precision matrix of a Gaussian distribution.

We denote the matrix representations of $\mathbf{u}_m$, $\mathbf{v}_n$, and $\mathbf{t}_k$ as $\mathbf{U} \equiv [\mathbf{u}_1, \mathbf{u}_2, \cdot\cdot, \mathbf{u}_M]$, $\mathbf{V} \equiv [\mathbf{v}_1, \mathbf{v}_2, \cdot\cdot, \mathbf{v}_N]$, and $\mathbf{T} \equiv [\mathbf{t}_1, \mathbf{t}_2, \cdot\cdot, \mathbf{t}_K]$. To account for randomness in ratings, BPTF uses the following probabilistic model for generating ratings:

$$\mathcal{R}|\mathbf{U}, \mathbf{V}, \mathbf{T} \sim \prod_{m=1}^{M} \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(\langle \mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k \rangle, \alpha^{-1}).$$

This represents the conditional distribution of $\mathcal{R}$ given $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{T}$ in terms of Gaussian distributions, each with means of $\langle \mathbf{u}_m, \mathbf{v}_n, \mathbf{t}_k \rangle$ and precision α.

The generative process of BPTF requires parameters $\boldsymbol{\mu}_0$, β_0, $\mathbf{W_0}$, ν_0, $\tilde{W}_0$, $\tilde{\Lambda}$, and $\tilde{\nu}_0$ in the hyper-priors, which should reflect prior knowledge about a specific problem and are treated as constants during training. The process is as follows:

1. Generate $\boldsymbol{\Lambda_U}$, $\boldsymbol{\Lambda_V}$, and $\boldsymbol{\Lambda_T} \sim \mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}_0, \nu_0)$, where $\boldsymbol{\Lambda_U}$, $\boldsymbol{\Lambda_V}$, and $\boldsymbol{\Lambda_T}$ are the precision matrices (a precision matrix is the inverse of a covariance matrix) for Gaussians. $\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}_0, \nu_0)$ is the Wishart distribution of a $D \times D$ random matrix $\boldsymbol{\Lambda}$ with ν_0 degrees of freedom and a $D \times D$ scale matrix $\mathbf{W_0}$:$\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W_0}, \nu_0) = \frac{|\boldsymbol{\Lambda}|^{(\nu_0 - \mathbf{D} - \mathbf{1})/\mathbf{2}}}{\mathbf{c}} \exp(-\frac{\mathbf{Tr}(\mathbf{W_0}^{-1}\boldsymbol{\Lambda})}{\mathbf{2}})$, where C is a constant.
2. Generate $\boldsymbol{\mu_U} \sim \mathcal{N}(\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda_U})^{-1})$, where $\boldsymbol{\mu_U}$ is used as the mean vector for a Gaussian. Similarly, generate $\boldsymbol{\mu_V} \sim \mathcal{N}(\boldsymbol{\mu}_0, (\beta_0\ \boldsymbol{\Lambda_V})^{-1})$ and $\boldsymbol{\mu_T} \sim \mathcal{N}(\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda_T})^{-1})$, where $\boldsymbol{\mu_V}$ and $\boldsymbol{\mu_T}$ are mean vectors for Gaussians.
3. Generate $\alpha \sim \mathcal{W}(\tilde{\Lambda}|\tilde{W}_0, \tilde{\nu}_0)$.
4. For each $m \in (1 \ldots M)$, generate $\mathbf{u}_m \sim \mathcal{N}(\boldsymbol{\mu_U}, {\boldsymbol{\Lambda_U}}^{-1})$.
5. For each $n \in (1 \ldots N)$, generate $\mathbf{v}_n \sim \mathcal{N}(\boldsymbol{\mu_V}, {\boldsymbol{\Lambda_V}}^{-1})$.
6. For each $k \in (1 \ldots K)$, generate $\mathbf{t}_k \sim \mathcal{N}(\boldsymbol{\mu_T}, {\boldsymbol{\Lambda_T}}^{-1})$.
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 $\boldsymbol{\mu}_0$, β_0, $\mathbf{W_0}$, ν_0, $\tilde{W}_0$, $\tilde{\Lambda}$, and $\tilde{\nu}_0$ should be set properly according to the objective dataset; fortunately, varying their values, has little impact on the final prediction [20].

BPTF views the hyper-parameters α, $\Theta_{\mathbf{U}} \equiv \{\boldsymbol{\mu_U}, \boldsymbol{\Lambda_U}\}$, $\Theta_{\mathbf{V}} \equiv \{\boldsymbol{\mu_V}, \boldsymbol{\Lambda_V}\}$, and $\Theta_{\mathbf{T}} \equiv \{\boldsymbol{\mu_T}, \boldsymbol{\Lambda_T}\}$ as random variables, yielding a predictive distribution for unobserved ratings $\hat{\mathcal{R}}$, which, for observable tensor $\mathcal{R}$, is given by:

$$p(\hat{\mathcal{R}}|\mathcal{R}) = \int p(\hat{\mathcal{R}}|\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha) \\ p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_{\mathbf{U}}, \Theta_{\mathbf{V}}, \Theta_{\mathbf{T}}|\mathcal{R}) d\{\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_{\mathbf{U}}, \Theta_{\mathbf{V}}, \Theta_{\mathbf{T}}\}. \quad (2)$$

BPTF computes the expectation of $p(\hat{\mathcal{R}}|\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha)$ over the posterior distribution $p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_{\mathbf{U}}, \Theta_{\mathbf{V}}, \Theta_{\mathbf{T}}|\mathcal{R})$; it approximates the expectation by averaging samples drawn from the posterior distribution. Since the posterior is too complex to be directly sampled, it applies the Markov Chain Monte Carlo (MCMC) indirect sampling technique to infer the predictive distribution for unobserved ratings $\hat{\mathcal{R}}$ (see [20] for details on the inference algorithm of BPTF).

The time and space complexities of BPTF are $O(\#nz \times D^2 + (M+N+K) \times D^3)$. $\#nz$ is the number of observation entries, and M, N, and K are all much greater than D. BPTF can also compute feature vectors in parallel while avoiding fine parameter tuning during factorization.

4 Method

We now explain S^{3}TF. We first explain how to create augmented tensors, which share semantics among services, from individual services' tensors. Table 1 summarizes the notations used by our method.

Table 1. Definition of main symbols.

Symbols	Definitions
$\mathcal{R}^i$	Tensor that includes ratings by users of venues with tags for the i-th service.
α^i	Observation precision for $\mathcal{R}^i$.
$\mathbf{u}^i_m$	m-th user feature vector for i-th service.
$\mathbf{v}^i_n$	n-th venue feature vector for i-th service.
$\mathbf{t}^i_k$	k-th tag feature vector for i-th service.
$\mathbf{U}^i$	Matrix representation of $\mathbf{u}^i_m$ for i-th service.
$\mathbf{V}^i$	Matrix representation of $\mathbf{v}^i_n$ for i-th service.
$\mathbf{T}^i$	Matrix representation of $\mathbf{t}^i_k$ for i-th service.
X	Number of services.
$\mathbb{V}^i_s$	Set of the most sparse venues for the i-th service.
$\mathbb{T}^i_s$	Set of the most sparse tags for the i-th service.
$\mathcal{A}^v$	The augmented tensor that includes the classes of sparse venues in all services.
$\mathcal{A}^t$	The augmented tensor that includes classes of sparse tags in all services.
$\mathbf{c}^v_j$	j-th semantically biased venue feature vector from $\mathcal{A}^v$.
$\mathbf{c}^t_j$	j-th semantically biased tag feature vector from $\mathcal{A}^t$.
$\mathbf{C}^v$	Matrix representation of $\mathbf{c}^v_j$.
$\mathbf{C}^t$	Matrix representation of $\mathbf{c}^t_j$.
S^v	Number of classes that include sparse venues in all services.
S^t	Number of classes that include sparse tags in all services.
$f(o)$	Function that returns the classes of object o.
δ	Parameter that adjusts the number of the most sparsely observed objects in each service.

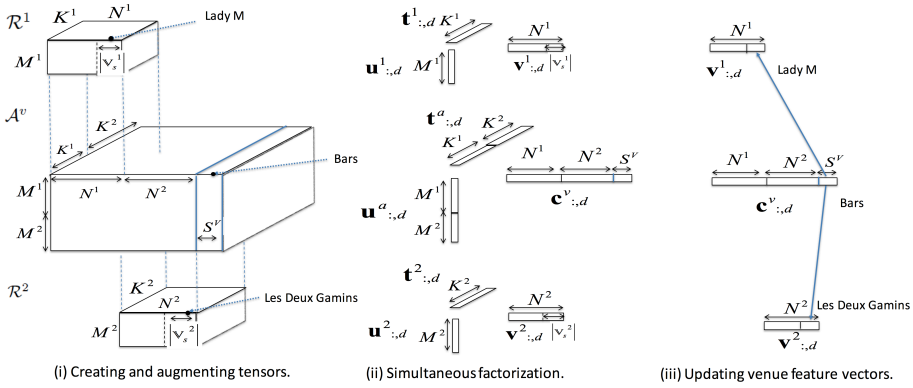


Fig. 2. Examples of our factorization process.

4.1 Creating augmented tensors

Following SSTF, S^{3}TF creates the augmented tensor $\mathcal{A}^v$ that has all the observations across X services (those services do not share any object) as well as the observations for sparsely observed venues lifted in the augmented venue classes. The classes are chosen from shared KBs such as DBPedia and Freebase, and thus they are shared among services; e.g. for restaurant review services, the types of restaurants and the food categories are listed in DBPedia or Freebase in detail.

First, S^{3}TF extracts the observations for sparsely observed venues. Here, the set of sparse venues for the i-th service ($1 \leq i \leq X$), denoted as $\mathbb{V}^i_s$, is defined as the group of the most sparsely observed venues, v^i_ss, among all venues in the i-th service. We set a 0/1 flag to indicate the existence of relationships composed of user u^i_m, venue v^i_n, and tag t^i_k as $o^i_{m,n,k}$. Then, $\mathbb{V}^i_s$ is computed as follows:

(1) S^{3}TF first sorts the venues from the rarest to the most common in the i-th service ($1 \leq i \leq X$) and creates a list of venues: $\{v^i_{s(1)}, v^i_{s(2)}, \ldots, v^i_{s(N^i-1)}, v^i_{s(N^i)}\}$ where N^i is the number of venues in the i-th service. For example, $v^i_{s(2)}$ is not less sparsely observed than $v^i_{s(1)}$.

(2) It iterates the following step (3) from $j = 1$ to $j = N^i$.

(3) If it satisfies the following equation, S^{3}TF adds the j-th sparse venue $v^i_{s(j)}$ to set $\mathbb{V}^i_s$: $(|\mathbb{V}^i_s| / \sum_{m,n,k} o^i_{m,n,k}) < \delta$ where $\mathbb{V}^i_s$ initially does not have any venues and $|\mathbb{V}^i_s|$ is the number of venues in set $\mathbb{V}^i_s$. If not, it stops the iterations and returns the set $\mathbb{V}^i_s$ as the most sparsely observed venues in the i-th service. Here, δ is a parameter used to determine the number of sparse venues in $\mathbb{V}^i_s$. Typically, we set δ to range from 0.05 to 0.20 in accordance with the long-tail characteristic such that sparse venues account for 5-20% of all observations [13].

Second, S^{3}TF constructs the augmented tensor $\mathcal{A}^v$ as follows:.

(1) S^{3}TF inserts the multi-object relationship composed of user u^i_m, venue v^i_n, and tag t^i_k, observed in the i-th service, into $\mathcal{A}^v$. Here, the rating $r^i{}_{m,n,k}$ corresponding to the above relationship is inserted into the $((M^{i-1}_1 + m), (N^{i-1}_1 + n), (K^{i-1}_1 + k))$-th element in $\mathcal{A}^v$ where we denote M^{i-1}_1, N^{i-1}_1, and K^{i-1}_1 as the sum of number of users, that of venues, and that of tags in services whose identifiers are from 1 to $(i-1)$, respectively. As a result, $\mathcal{A}^v$ has all users, all venues, and all tags in all services. In Fig. 2-(i), all observations in $\mathcal{R}^1$ and $\mathcal{R}^2$ are inserted into $\mathcal{A}^v$.

(2) S^{3}TF additionally inserts the multi-object relationships composed of user u^i_m, a class of sparse venue c^v_j, and tag t^i_k into $\mathcal{A}^v$ if v^i_n is included in $\mathbb{V}^i_s$ and c^v_j is one of the classes of v^i_n. Thus, the rating $r^i{}_{m,n,k}$ is inserted into the $((M^{i-1}_1 + m), (N^X_1 + j), (K^{i-1}_1 + k))$-th element in $\mathcal{A}^v$. If sparse venue v^i_n has several classes, S^{3}TF inserts the rating $r^i{}_{m,n,k}$ into all corresponding elements in $\mathcal{A}^v$. In Fig. 2-(i), observations for classes for sparse venues ("Lady M" in service 1 and "Les Deux Gamins" in service 2) are added to $\mathcal{A}^v$ (in the elements corresponding to their class "Bars"). Here, the number of classes that have the sparse venues in all services is denoted as S^v; it is computed as: $S^v = |\bigcup_{\mathbb{V}^i_s} f(v^i_s)|_{(1 \leq i \leq X)}$ where $f(v^i_s)$ is a function that returns the classes of sparse venue v^i_s in the i-th service.

The set of sparse tags $\mathbb{T}^i_s$ is defined as the group of the most sparsely observed tags in i-th service and is computed using the same procedure as it creates $\mathbb{V}^i_s$. The augmented tensor for tags $\mathcal{A}^t$ is also computed in the same way as it creates $\mathcal{A}^v$. So we omit the explanations of the procedures for creating those here.

Tensor creation by S^{3}TF has the following two benefits: (1) It solves the balance problem by creating individual tensors for services and so avoids strongly biasing any particular service. (2) It overcomes the sparsity problem by propagating observations in sparse objects to their classes shared among services in the augmented tensor.

4.2 Simultaneously factorizing tensors across services

S^{3}TF factorizes individual services' tensors and augmented tensors simultaneously. We first explain our approach and then the algorithm.

Approach S^{3}TF takes the following three techniques in factorizing tensors.

(A) It factorizes individual service tensors $\mathcal{R}^i$s ($1 \leq i \leq X$), and augmented tensors $\mathcal{A}^v$ and $\mathcal{A}^t$ simultaneously. In particular, it creates feature vectors for users, $\mathbf{u}^i_m$s, those for venues, $\mathbf{v}^i_n$s, and those for tags, $\mathbf{c}^t_j$s, by factorizing tensor $\mathcal{R}^i$ for each i-th service as well as feature vectors for their venue classes $\mathbf{c}^v_j$s by $\mathcal{A}^v$ and those for their tag classes $\mathbf{c}^t_j$s by $\mathcal{A}^t$. As a result, S^{3}TF factorizes individual tensors while enabling the semantic biases from $\mathbf{c}^v_j$s and $\mathbf{c}^t_j$s to be shared during the factorization process. This approach to "simultaneously" factorizing individual service tensors solves the balance problem. In the example shown in Fig. 2-(ii), $\mathcal{R}^1$, $\mathcal{R}^2$, and $\mathcal{A}^v$ are factorized simultaneously into D-dimensional "row" feature vectors.

(B) It shares feature vectors $\mathbf{u}^i_m$, $\mathbf{v}^i_n$, $\mathbf{t}^i_k$ which are computed by factorizing $\mathcal{R}^i$, in the factorization of augmented tensors $\mathcal{A}^v$ and $\mathcal{A}^t$. This means that it computes the feature matrix for users for the augmented tensor $\mathbf{U}^a$ by joining X numbers of service feature matrices for users, $[\mathbf{U}^1, \ldots, \mathbf{U}^i, \ldots, \mathbf{U}^X]$. Similarly, it computes the feature matrix for venues, $\mathbf{V}^a$, and that for tags, $\mathbf{T}^a$, for the augmented tensor. Then, it computes the feature matrix for venue (or tag) classes by reusing the joined feature matrices $\mathbf{U}^a$ and $\mathbf{T}^a$ (or $\mathbf{U}^a$ and $\mathbf{V}^a$). As a result, it can, during the factorization process, share the tendencies of users' activities across services via those shared parameters. In Fig. 2-(ii), $\mathbf{u}^a_{m,d}$ is computed as: $[\mathbf{u}^1_{m,d}, \mathbf{u}^2_{m,d}]$ and $\mathbf{t}^a_{k,d}$ is as: $[\mathbf{t}^1_{k,d}, \mathbf{t}^2_{k,d}]$.

(C) It updates latent feature vectors for sparse venues (or tags) in the i-th service, $\mathbf{v}^i_s$s (or $\mathbf{t}^i_s$s), by incorporating semantic biases from $\mathbf{c}^v_j$s (or $\mathbf{c}^t_j$s) to $\mathbf{v}^i_s$s (or $\mathbf{t}^i_s$s). Here, $\mathbf{c}^v_j$s (or $\mathbf{c}^t_j$s) are feature vectors for classes of the sparse venues v^i_ss (or sparse tags t^i_ss). This process incorporates the semantic tendencies of users' activities across services captured by idea (B) into each service's factorization; this is useful in solving the sparsity problem. In Fig. 2-(iii), each row vector $\mathbf{c}^v_{:,d}$ has latent features for $(N^1 + N^2)$ venues and for S^v classes. The features in $\mathbf{c}^v_{:,d}$ share semantic knowledge of sparse venues across services. For example, the feature for "Bars" in $\mathbf{c}^v_{:,d}$ share semantic knowledge of sparse venues "Lady M" and "Les Deux Gamins" across US restaurant review service and French one (see also Fig. 1).

Algorithm Here we explain how to compute the predictive distribution for unobserved ratings. Differently from the BPTF model (see Eq. (2)), S^{3}TF considers the tensors for individual services and augmented tensors in computing the distribution. Thus, the predictive distribution is computed as follows:

$$
\begin{aligned}
p(\hat{\mathcal{R}}|\mathcal{R}, \mathcal{A}^v, \mathcal{A}^t) = \int & p(\hat{\mathcal{R}}|\mathbf{U}, \mathbf{V}, \mathbf{T}, \mathbf{C}^v, \mathbf{C}^t, \alpha, \alpha^a) \\
& p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \mathbf{C}^v, \mathbf{C}^t, \Theta_{\mathbf{U}}, \Theta_{\mathbf{V}}, \Theta_{\mathbf{C}^v}, \Theta_{\mathbf{C}^t}, \alpha, \alpha^a|\mathcal{R}, \mathcal{A}^v, \mathcal{A}^t) \\
& d\{\mathbf{U}, \mathbf{V}, \mathbf{T}, \mathbf{C}^v, \mathbf{C}^t, \Theta_{\mathbf{U}}, \Theta_{\mathbf{V}}, \Theta_{\mathbf{T}}, \Theta_{\mathbf{C}^v}, \Theta_{\mathbf{C}^t}, \alpha, \alpha^a\} \quad (3)
\end{aligned}
$$

where $\mathcal{R} \equiv \{\mathcal{R}^i\}_{i=1}^X$, $\alpha \equiv \{\alpha^i\}_{i=1}^X$, $\mathbf{U} \equiv \{\mathbf{U}^i\}_{i=1}^X$, $\mathbf{V} \equiv \{\mathbf{V}^i\}_{i=1}^X$, $\mathbf{T} \equiv \{\mathbf{T}^i\}_{i=1}^X$, $\Theta_{\mathbf{U}} \equiv \{\Theta_{\mathbf{U}^i}\}_{i=1}^X$, $\Theta_{\mathbf{V}} \equiv \{\Theta_{\mathbf{V}^i}\}_{i=1}^X$, and $\Theta_{\mathbf{T}} \equiv \{\Theta_{\mathbf{T}^i}\}_{i=1}^X$.

Eq. (3) involves a multi-dimensional integral that cannot be computed analytically. Thus, S^{3}TF views Eq. (3) as the expectation of $p(\hat{\mathcal{R}}|\mathcal{R},\mathcal{A}^v,\mathcal{A}^t)$ over the posterior distribution $p(\mathbf{U},\mathbf{V},\mathbf{T},\mathbf{C}^v,\mathbf{C}^t,\Theta_{\mathbf{U}},\Theta_{\mathbf{V}},\Theta_{\mathbf{C}^v},\Theta_{\mathbf{C}^t},\alpha,\alpha^a|\mathcal{R},\mathcal{A}^v,\mathcal{A}^t)$, and approximates the expectation by MCMC with the Gibbs sampling paradigm. It collects a number of samples, L, to approximate the integral in Eq. (3) as:

$$p(\hat{\mathcal{R}}|\mathcal{R},\mathcal{A}^v,\mathcal{A}^t) \approx \sum_{l=1}^{L} p(\hat{\mathcal{R}}|\mathbf{U}[l],\mathbf{V}[l],\mathbf{T}[l],\mathbf{C}^v[l],\mathbf{C}^t[l],\alpha[l],\alpha^a[l]) \tag{4}$$

where l represents the l-th sample.

The MCMC procedure is as follows (detail is given in the supplemental material[3]):

(1) Initialize $\mathbf{U}^i[1]$, $\mathbf{V}^i[1]$, and $\mathbf{T}^i[1]$ $(1 \leq i \leq X)$ for each i-th service as well as $\mathbf{C}^v[1]$ and $\mathbf{C}^t[1]$ for the augmented tensors by Gaussian distribution as per BPTF. $\mathbf{C}^v[1]$ and $\mathbf{C}^t[1]$ are used for sharing the semantics across services (see our approach (A)). Next, it repeats steps (2) to (8) L times.

(2) Samples the hyperparameters for each i-th service as per BPTF i.e.:
- $\alpha^i[l{+}1] \sim p(\alpha^i[l]|\mathbf{U}^i[l],\mathbf{V}^i[l],\mathbf{T}^i[l],\mathcal{R}^i)$
- $\Theta_{U^i}[l{+}1] \sim p(\Theta_{U^i}[l]|\mathbf{U}^i[l])$
- $\Theta_{V^i}[l{+}1] \sim p(\Theta_{V^i}[l]|\mathbf{V}^i[l])$
- $\Theta_{T^i}[l{+}1] \sim p(\Theta_{T^i}[l]|\mathbf{T}^i[l])$

here, $\Theta_{\mathbf{X}} \equiv \{\mu_{\mathbf{X}},\Lambda_{\mathbf{X}}\}$ and is computed in the same way as BPTF.

(3) Samples the feature vectors the same way as is done in BPTF:
- $\mathbf{u}_m^i[l{+}1] \sim p(\mathbf{u}_m^i|\mathbf{V}^i[l],\mathbf{T}^i[l],\alpha^i[l{+}1],\Theta_{U^i}[l{+}1],\mathcal{R}^i)$
- $\mathbf{v}_n^i[l{+}1] \sim p(\mathbf{v}_n^i|\mathbf{U}^i[l{+}1],\mathbf{T}^i[l],\alpha^i[l{+}1],\Theta_{V^i}[l{+}1],\mathcal{R}^i)$
- $\mathbf{t}_k^i[l{+}1] \sim p(\mathbf{t}_k^i|\mathbf{U}^i[l{+}1],\mathbf{V}^i[l{+}1],\alpha^i[l{+}1],\Theta_{T^i}[l{+}1],\mathcal{R}^i)$

(4) Joins the feature matrices in services in order to reuse them as the feature matrices for the augmented tensors as (see our approach (B)):
- $\mathbf{U}^a[l{+}1] = [\mathbf{U}^1[l{+}1],\cdots,\mathbf{U}^i[l{+}1],\cdots,\mathbf{U}^X[l{+}1]]$
- $\mathbf{V}^a[l{+}1] = [\mathbf{V}^1[l{+}1],\cdots,\mathbf{V}^i[l{+}1],\cdots,\mathbf{V}^X[l{+}1]]$
- $\mathbf{T}^a[l{+}1] = [\mathbf{T}^1[l{+}1],\cdots,\mathbf{T}^i[l{+}1],\cdots,\mathbf{T}^X[l{+}1]]$

(5) Samples the hyperparameters for the augmented tensors similarly:
- $\alpha^a[l{+}1] \sim p(\alpha^a[l]|\mathbf{U}^a[l{+}1],\mathbf{V}^a[l{+}1],\mathbf{T}^a[l{+}1],\mathcal{R}^a)$
- $\Theta_{C^v}[l{+}1] \sim p(\Theta_{C^v}[l]|\mathbf{C}^v[l])$
- $\Theta_{C^t}[l{+}1] \sim p(\Theta_{C^t}[l]|\mathbf{C}^t[l])$

(6) Samples the semantically-biased feature vectors by using $\alpha^a[l{+}1]$, $\mathbf{U}^a[l{+}1]$, $\mathbf{V}^a[l{+}1]$, and $\mathbf{T}^a[l{+}1]$ as follows (see our approach (B)):
- $\mathbf{c}_j^v[l{+}1] \sim p(\mathbf{c}_j^v|\mathbf{U}^a[l{+}1],\mathbf{T}^a[l{+}1],\alpha^a[l{+}1],\Theta_{C^v}[l{+}1],\mathcal{A}^v)$
- $\mathbf{c}_j^t[l{+}1] \sim p(\mathbf{c}_j^t|\mathbf{U}^a[l{+}1],\mathbf{V}^a[l{+}1],\alpha^a[l{+}1],\Theta_{C^t}[l{+}1],\mathcal{A}^t)$

(7) Samples the unobserved ratings $\hat{r}_{m,n,k}^i[l{+}1]$ by applying $\mathbf{U}^i[l{+}1]$, $\mathbf{V}^i[l{+}1]$, $\mathbf{T}^i[l{+}1]$, $\mathbf{C}^v[l{+}1]$, $\mathbf{C}^t[l{+}1]$, $\alpha^i[l{+}1]$ to equation (4).

[3] Please see "https://sites.google.com/site/sapplementalfile/appendix-html".

(8) Updates $\mathbf{v}_n^i[l{+}1]$ as follows and uses it in the next iteration (see our approach (C)):

$$\mathbf{v}_n^i \quad = \quad \begin{cases} \frac{1}{2}\left(\mathbf{v}_n^i + \frac{\sum_{c_j^v \in f(v_n^i)} \mathbf{c}_j^v}{|f(v_n^i)|}\right) & (v_n^i \in \mathbb{V}_s^i) \\ \mathbf{v}_n^i & (\text{otherwise}) \end{cases} \tag{5}$$

Updates $\mathbf{t}_k^i[l{+}1]$ similarly (we halt the explanation here).

The complexity of S^{3}TF in each MCMC iteration is $O(\#nz \times D^2 + (M_1^X + N_1^X + K_1^X + S^V + S^T) \times D^3)$. Because the first term is much larger than the rest, the computation time is almost the same as that of BPTF. Parameter δ and parameters for factorization can be easily set based on the long-tail characteristic and the full Bayesian treatment inherited by the BPTF framework, respectively. S^{3}TF is faster than SSTF when analyzing X numbers of services since S^{3}TF creates and factorizes only one set of augmented tensors ($\mathcal{A}^v$ and $\mathcal{A}^t$) for all services while SSTF needs X sets of augmented tensors.

5 Evaluation

The method's accuracy was confirmed by evaluations.

5.1 Dataset

We used the Yelp ratings/reviews[4] together with DBPedia [2] food vocabularies. Yelp datasets contain user-made ratings of restaurant venues and user reviews of venues across four countries (United Kingdom (UK), United States (US)[5], Canada[6], and Germany). The logs of users who are included in several countries are excluded from the datasets. Thus we can consider the datasets of individual countries are made from different services. Food vocabularies are extracted from food ontology[7] and categories are extracted from DBPedia article categories. We first extracted English food entries and then translated them into French or German by using BabelNet[8], which is a multilingual encyclopedic dictionary based on Wikipedia entries. Thus, the resulting food entries share the same categories. We then extracted tags from the reviews that match the instances in a DBPedia food vocabulary entry as was done in [13]. Consequently, we extracted 988, 1,100, 1,388, and 435 tags for UK, US, Canada, Germany, respectively. We used the genre vocabulary provided by Yelp as the venue vocabulary, it has 179 venue classes. The tag vocabulary provided by DBPedia has 1,358 food classes.

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

[5] We focused on restauran reviews for Midwestern United States to efficiently perform evaluations.

[6] The Canada dataset includes venues located in the Quebec area, so the languages used in the reviews are written in French or English.

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

[8] http://babelnet.org

The size of the user-venue-tag tensors in UK, US, Canada, and Germany were $2,052\times1,398\times988$, $10,736\times1,554\times1,100$, $10,700\times3,085\times1,388$, and $286\times332\times435$, respectively. The numbers of ratings in those countries were 54,774, 118,012, 172,182, and 3,062, respectively. The ratings range from 1 to 5.

5.2 Comparison methods

We compared the accuracy of the following six methods:

1. *NMTF* [19], which utilizes the auxiliary information like GCTF. It factorizes the target tensors (user-item-tag tensors created for each countries) and auxiliary matrices (item-class matrix and tag-class matrix) simultaneously.
2. *BPTF* proposed by [20].
3. *SSTF*, which applies Semantic Sensitive Tensor Factorization proposed by [13] to the observed relationships in each service.
4. *SSTF_all*, which combines observed relationships in different services to create a merged tensor and factorizes the merged tensor by SSTF.
5. S^3*TFT*, which utilizes only the tag vocabulary.
6. S^3*TFV*, which utilizes only the venue vocabulary.
7. S^3*TF*, which is our proposal.

5.3 Methodology and parameter setup

We split each dataset into two halves; a training set that holds reviews entered in the first half period of all logs and a test set consisting of the reviews entered in the last half. We then performed evaluations for the two-joint combinations (total 6) of those sets to check the repeatability of results. Following the evaluation methodology used in previous studies [3, 11, 13, 20], we computed Root Mean Square Error (RMSE), which is computed by $\sqrt{(\sum_{i=1}^{n}(P_i - R_i)^2)/n}$, where n is the number of entries in the test set, and P_i and R_i are the predicted and actual ratings of the i-th entry, respectively. The RMSE is more appropriate to represent model performance than the Mean Absolute Error (MAE) when the error distribution is expected to be Gaussian. We varied D from 5 to 20 for each method, and set the optimum value to 20 since it gave the highest accuracies for all methods. We set the iteration count, L, to 100 since all methods could converge with this setting. δ was set to 0.8 following [13].

5.4 Results

We first investigated the sparseness of objects observed. Fig. 3 plots the distribution of venue frequencies observed in the UK dataset. From this figure, we can confirm that venue observation frequencies exhibit the long-tail characteristic. Thus, observations of multi-object relationships become very sparse with respect to the possible combinations of observed objects. The distributions of other datasets showed the same tendencies. Thus, a solution to the sparsity problem across services is required. Fig. 4 presents the accuracy of the UK dataset

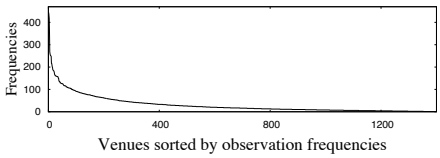


Fig. 3. Distribution of venue frequencies.

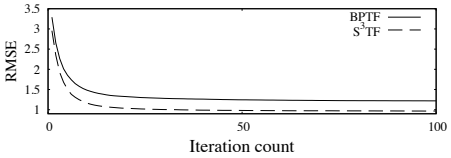


Fig. 4. Accuracy vs. iteration count.

Table 2. Comparing RMSE values of the methods.

	NMTF	*BPTF*	*SSTF*	*SSTF_all*	S^3TFT	S^3TFV	S^3TF
UK	1.7192	1.0063	0.9928	0.9960	0.9967	0.9594	***0.9501***
US	1.9011	1.2303	1.2176	1.2267	1.1939	1.1733	***1.1727***
Canada	1.8723	1.1853	1.1431	1.1655	1.1219	1.1331	***1.1215***
German	1.8923	1.3266	1.2789	1.2868	1.2744	1.2847	***1.2527***

on the simultaneous factorization on UK and US datasets when the number of iterations, L, was changed. This confirms that the accuracy of S^3TF saturated before L=100. Results on other datasets showed similar tendencies.

We then compared the accuracy of the methods for the simultaneous factorizations on the six datasets. The results shown in Table 2 are the average RMSE values computed for each country. They show that *SSFT* has better accuracy than *BPFT*. This is because *SSFT* uses the semantics shared within a single service (e.g. within a service in US) and thus solves the sparsity problem. *SSTF* has better accuracy by *SSTF_all* though *SSTF_all* uses the entire logs. This is because *SSTF_all* creates a tensor by mixing the heterogeneous datasets in different countries and thus suffers from the balance problem. S^3TFT and S^3TFV had better performance than *BPTF* or *SSTF* since S^3TFT and S^3TFV can use the shared semantics on venues and those on tags across services, respectively. Finally, S^3TF, which utilizes the semantic knowledge across services while performing coupled analysis of two tensors, yielded higher accuracy than the current best method, *SSTF*, with the statistical significance of α<0.05.

The RMSEs of *NMTF* are much worse than those of S^3TF. This is mainly because: (1) *NMTF* straightforwardly combines different relationships, i.e., rating relationships among users, items, and tags, link relationships among items and their classes, and link relationships among tags and their classes. Thus, it suffers from the balance problem. (2) *NMTF* uses the KL divergence for optimizing the predictions since its authors are interested in "discrete value observations such as stars in product reviews", as described in [19]. Our datasets are those they are interested in; however, exponential family distributions like Poisson distribution do not fit our rating datasets so well.

Table 3 presents the computation times of *BPTF*, *SSTF*, and S^3TF when simultaneously factorizing tensor for German and that for UK datasets as well as simultaneously factorizing tensor for German and that for US datasets. All experiments were conducted on a Linux 3.33 GHz Intel Xeon (24 cores) server with 192 GB of main memory. All methods were implemented with Matlab and GCC. We can see that the computation time of S^3TF is shorter than that of *SSTF*. Furthermore, we can set L smaller than 100 (see Fig. 4). Thus, we can

Table 3. Computation time (seconds) when L=100.

German x UK			German x US		
$BPTF$	$SSTF$	S^3TF	$BPTF$	$SSTF$	S^3TF
63	113	109	85	142	134

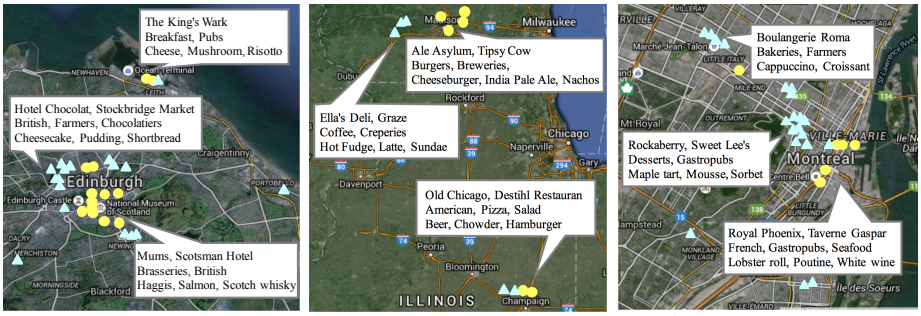
Table 4. Prediction examples for US (the upper row) and German (the lower row).

Training dataset			Rating predictions by $SSTF$ (S) and S^3TF (S^3)				
Tag in review sentence	Item/genre	Rating	Tag in review sentence	Item/genre	S	S^3	Actual
Berry ***streusel*** is tasty.	A/Bakeries	5.0	***Bratwurst*** was incredible.	B/American	3.8	4.7	5.0
A ***enchilada*** is perfect.	C/Tex-Mex	4.0	An amazing ***pretzel*** roll.	D/Breakfast	3.8	4.8	5.0
Ich genoss ***Marzipan***.	E/Bakeries	5.0	***Burrito*** war wirklich gut.	F/Bars	3.1	3.7	4.0
nachos ist wertvoll.	G/Bars	4.0	***Schnitzel*** ist lecker.	H/German	3.9	3.4	3.0

conclude that S^3TF can compute more accurate predictions quickly; it works better than $SSTF$ and $BPTF$ on real applications.

We then show the examples of the differences between the predictions output by $SSTF$ and S^3TF in Table 4. The columns "S", "S^3", and "Actual" list prediction values by $SSTF$, those by S^3TF, and actual ratings given by users as found in the test dataset, respectively. In the US dataset, the combination of tag "streusel" at Bakeries "A" and "enchilada" at Tex-Mex restaurant "C" were highly rated in the training set. In the test set, the combination of tag "bratwurst" at American restaurant "B" and "pretzel" at Breakfast restaurant "D" were highly rated. The tags "streusel", "bratwurst", and "pretzel" (they are included in "german cuisine" class) are sparsely observed in the US's training set. In the Germany dataset, the combination of tag "marzipan" at Bakeries "E" and "nachos" at Bars "G" were highly rated in the training set. In the test set, the combination of tag "burrito" at Bars "F" and "Schnitzel" at German restaurant "H" were highly rated. The tags "nachos" and "burrito" (they are included in "mexican cuisine" class) are sparsely observed in the German's training set. S^3TF accurately predicted those observations formed by sparse tags since it uses knowledge that the tags "streusel" and "marzipan" both lie in tag class "german cuisine", as well as the knowledge that tags "enchilada" and "nachos" both lie in tag class "mexican cuisine". Thus, S^3TF can use such knowledge that the combinations of "german cuisine" and "mexican cuisine" are often seen in datasets across countries. $SSTF$ predictions were inaccurate since they were not based on the semantics behind the objects being rated across services.

We also show the implicit relationships extracted when we factorized three datasets, UK, US, and Canada, simultaneously. The implicit relationships were computed as: (1) The probability that the relationship composed by u_m, v_n, and t_k is included in the i-th dimension is computed by $u_{i,m} \cdot v_{i,n} \cdot t_{i,k}$ where $1 \leq i \leq D$. (2) Each observed relationship is classified into the dimension that gives the highest probability value among all D dimensions. (3) The relationships included in the same dimension are considered to form implicit relationships across services. Fig. 5 presents examples as the extraction results. The first line, the second line, and the third line in balloons in the figure indicate the representative venues, venue classes, and foods, respectively. The relationships in the same dimension are represented by the same marks; circles (1) or triangles



(a) Relationships in UK (b) Relationships in US (c) Relationships in Canada

Fig. 5. Examples of implicit relationships extracted by S^{3}TF.

(2): (1) This dimension includes several local dishes with alchoholic content across countries. e.g. People in UK who love "Haggis" and drink "Scotch whisky" are implicitly related to those in US who love "Cheeseburger" and drink "Indian pale ale" as well as those in Canada who love "Lobster roll" and drink "White wine". (2) This dimension includes several sweet dishes across countries. e.g. People in UK who love "Shortbread" are implicitly related to those in US who love "Sundae" as well as those in Canada who love "Maple tart". Such implicit relationships can be used to create recommendation lists for users across services. BPTF and SSTF can not extract such implicit relationships since they can not use shared semantics, and thus latent features, across services.

6 Conclusion

This is the first study to show how to include the semantics behind objects into tensor factorization and thus analyze users' activities across different services. Semantic-Sensitive Simultaneous Tensor Factorization, S^{3}TF, proposed here, presents a new research direction to the use of shared semantics for the cross service analysis of users' activities. S^{3}TF creates individual tensors for different services and links the objects observed in each tensor to the shared semantics. Then, it factorizes the tensors simultaneously while integrating semantic biases into tensor factorization. Experiments using real-world datasets showed that S^{3}TF achieves much higher accuracy than the current best tensor method and extracts implicit relationships across services during factorization. One interesting future direction is to apply our idea to the recent embedding models (e.g. [23]) and analyze different services simultaneously by using KBs.

References

1. Bizer, C., Heath, T. and Berners-Lee, T.: Linked data - the story so far, *Int. J. Semantic Web Inf. Syst.*, Vol. 5, No. 3, p. 122 (2009).
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S.: DBpedia - A crystallization point for the Web of Data, *Journal of Web Semantics*, Vol. 7, No. 3, pp. 154–165 (2009).

3. Cemgil, A. T.: Bayesian Inference for Nonnegative Matrix Factorisation Models, *Computational Intelligence and Neuroscience*, Vol. 2009, pp. 4:1–4:17 (2009).
4. Deshpande, O., Lamba, D. S., Tourn, M., Das, S., Subramaniam, S., Rajaraman, A., Harinarayan, V. and Doan, A.: Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches, *Proc. SIGMOD'13*, pp. 1209–1220 (2013).
5. http://www.ntt.co.jp/csr_e/2013report/pdf/en_19-22.pdf
6. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S. and Zhang, W.: Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion, *Proc. KDD'14*, pp. 601–610 (2014).
7. Franz, T., Schultz, A., Sizov, S. and Staab, S.: TripleRank: Ranking Semantic Web Data by Tensor Decomposition, *Proc. ISWC'09*, pp. 213–228 (2009).
8. Hoffart, J., Suchanek, F. M., Berberich, K. and Weikum, G.: YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia, *Artif. Intell.*, Vol. 194, pp. 28–61 (2013).
9. Karatzoglou, A., Amatriain, X., Baltrunas, L. and Oliver, N.: Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, *Proc. RecSys'10* pp. 79–86 (2010).
10. Menon, A. K. and Elkan, C.: Link Prediction via Matrix Factorization, *Proc. ECML/PKDD'11*, pp. 437–452 (2011).
11. Nakatsuji, M., Fujiwara, Y., Toda, H., Sawada, H., Zheng, J. and Hendler, J. A.: Semantic Data Representation for Improving Tensor Factorization, *Proc. AAAI'14*, pp. 2004–2012 (2014).
12. Nakatsuji, M., Fujiwara, Y., Uchiyama, T. and Toda, H.: Collaborative filtering by analyzing dynamic user interests modeled by taxonomy, *Proc. ISWC'12*, pp. 361–377 (2012).
13. Nakatsuji, M., Toda, H., Sawada, H., Zheng, J. G. and Hendler, J. A.: Semantic sensitive tensor factorization, *Artif. Intell.*, Vol. 230, pp. 224–245 (2016).
14. Nickel, M., Tresp, V. and Kriegel, H.-P.: Factorizing YAGO: scalable machine learning for linked data, *Proc. WWW'12*, pp. 271–280 (2012).
15. Niu, F., Zhang, C., R, C. and Shavlik, J. W.: Elementary: Large-Scale Knowledge-Base Construction via Machine Learning and Statistical Inference., *Int. J. Semantic Web Inf. Syst.*, Vol. 8, No. 3, pp. 42–73 (2012).
16. Pirr, G.: Explaining and Suggesting Relatedness in Knowledge Graphs., *Proc. ISWC'15*, pp. 622–639 (2015).
17. Rendle, S. and Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation, *Proc. WSDM'10*, pp. 81–90 (2010).
18. Rowe, M.: Transferring Semantic Categories with Vertex Kernels: Recommendations with SemanticSVD++, *Proc. ISWC'14*, pp. 341–356 (2014).
19. Takeuchi, K., Tomioka, R., Ishiguro, K., Kimura, A. and Sawada, H.: Non-negative Multiple Tensor Factorization, *Proc. ICDM'13*, pp. 1199–1204 (2013).
20. Xiong, L., Chen, X., Huang, T.-K., Schneider, J. G. and Carbonell, J. G.: Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization, *Proc. SDM'2010*, pp. 211–222 (2010).
21. Xu, M., Wang, Z., Bie, R., Li, J., Zheng, C., Ke, W. and Zhou, M.: Discovering Missing Semantic Relations between Entities in Wikipedia, *Proc. ISWC'13*, pp. 657–670 (2013).
22. Yilmaz, Y. K., Cemgil, A.-T. and Simsekli, U.: Generalised Coupled Tensor Factorisation, *Proc. NIPS'11* (2011).
23. Zhang, F., Yuan, N. J., Lian, D., Xie, X. and Ma, W.-Y.: Collaborative Knowledge Base Embedding for Recommender Systems, *Proc. KDD'16* (2016).