Context-Aware Point of Interest Recommendation using Tensor Factorization

Stathis Maroulis, Ioannis Boutsis, Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business
Athens, Greece
Email: {maroulise,mpoutsis,vana}@aueb.gr

***Abstract*—The wide adoption of Location Based Social Networks along with advances in mobile technology, has brought forth as a core service the analysis of large volumes of location-based data for personalized Point of Interest (POIs) recommendations. The majority of the existing recommendation systems take advantage of Collaborative Filtering, but they fail to exploit the contextual information involved with POI checkins (*i.e.,*, POI category, location, or the checkin timestamp). In this paper we propose CoTF, a Context-Aware Point of Interest Recommendation system using Tensor Factorization, that aims at enhancing the user experience by providing personalized context aware POI recommendations. Our approach exploits Category-based context related to checkins without the need of any pre- or post- filtering techniques. Our detailed experimental evaluation using real data from the Foursquare location-based social network illustrates that our approach can efficiently produce personalized recommendations to users, while significantly reducing the training time compared to current state-of-the-art methods.**

***Index Terms*—Tensor Factorization, Point of Interest Recommendation, Contextual Recommendation**

I. Introduction

Over the last decade, the wide availability of mobile devices (*e.g.*, smartphones, tablets, etc.) equipped with various types of sensors, including GPS, accelerometers and WiFi, along with advances in mobile networks, have led to the development of Location-Based Social Networks (LBSNs). LBSNs are social networks that incorporate the spatial dimension in the content shared by the users. For example, LBSNs such as Foursquare[1] and GeoLife [18] allow users to check-in into places, advertise these places to their friends or leave comments to share their experiences with their social circles. Introducing such user geographical locations in the social data enables us to retrieve rich information including user activities, mobility patterns or social structures. Such information has been recently exploited commercially to provide additional services to the users.

One of the most popular services based on this kind of information is to provide personalized Point of Interest (POI) recommendations to users in LBSNs in order to improve their experience. POI Recommendation systems use the information provided by LBSNs and generate personalized predictions for each user based on user behavior analysis. Recent schemes have been proposed for recommending items such as books, movies or even hashtags [9], [16]. However, the problem of recommending POIs in LBSN is much more challenging for a number of reasons: (i) POIs provide richer information that can be exploited to provide efficient recommendations, including POI category, spatial and temporal information from the user check-ins, (ii) user interests can vary significantly depending on time and location, (iii) user behaviors can be correlated spatio-temporally, and (iv) in contrast to traditional systems that exploit user ratings for recommendations, LBSNs only provide check-in information, and thus, there is no information about places that users dislike. Therefore, the development of POI recommendation system is much more complex than traditional recommendation systems.

Recent works that focus on POI Recommendation exploit Matrix Factorization using user check-ins to deal with the challenge of the extreme sparsity of user-POI matrices, as this approach serves collaborative filtering with implicit feedback better [6], [14]. However, approaches based on Matrix Factorization do not allow us to include context information such as POI category, check-in times, etc. As we show in this paper, such information can highly improve the recommendations, as they can reveal hidden relations between users and POIs (*e.g.*, if a user has never checked into a specific POI category, it is very likely that the user does not like this category). Other works have tried to exploit this information as shown in [12] by integrating the category of the POIs using pre-filtering and post-filtering approaches in Matrix Factorization. Recently, Lian *et al.* in [11] augmented user and POI latent factors of the factorization model with non-negative activity area vectors of users and non-negative influence area vectors of POIs to improve the accuracy of the POI recommendations. These works have the drawback of significantly higher complexity, and they also lack flexibility as they are able to cope with only one kind of context.

Our approach is based on the use of the Tensor Factorization technique for context-aware POI recommendation. Tensors are higher order matrices (*e.g.*, a third order tensor is a 3D-Matrix). Using tensors we can extend the classic user-POI matrix to a user-POI-context tensor, where each tuple indicates how many time a user has visited a POI at a specific context (*i.e.*, category, location or time). Tensor Factorization is a generalization of Matrix Factorization that provides a flexible and generic integration of contextual information without using any post or pre-filtering techniques that adds significant complexity to the models. Tensor Factorization techniques were commonly used for traditional recommendation systems with both explicit, and implicit feedback [5], [7]. We exploit

Tensor Factorization for POI recommendations. To the best of our knowledge this is the first work that uses Tensor Factorization for this setting.

In this paper we propose CoTF (Context-Aware Point of Interest Recommendation using Tensor Factorization), a context-aware POI recommendation system that aims at enhancing the user experience when discovering new places. CoTF determines the POIs that a user might want to visit, according to user preferences and contextual information. In this paper we focus on a context called category transition, but our model could easily handle more kinds of context, without modification. Thus, CoTF exploits check-in information from LBSNs and determines the unknown interdependencies among users and POIs, based on Tensor Factorization, to propose POIs that the users would like to visit. Furthermore, our solution employs a shuffling-based approach that allows the Tensor Factorization approach to converge to the solution significantly faster compared to traditional approaches [4]. Our experimental evaluation, using two datasets extracted from Foursquare, illustrates that our approach is efficient and practical, and makes accurate POI recommendations.

II. System Model

Location-based Social Networks (LBSNs). Location-based Social Networks (LBSNs) comprise individuals connected through inter-dependencies derived from their locations in the physical world and location-tagged media content, such as photos, video, and text. These inter-dependencies reveal not only that two persons co-exist in the same physical location or share similar locations, but also reveal information that can be inferred from users that visit similar locations (*e.g.*, common interests, behavior, and activities) [20], [21].

We define as Point of Interest (POI), denoted as i, a place that a human visits in a LBSN, such as museum, bar, stadium, etc. Each POI i is associated with a set of attributes: $\langle id_i, latitude_i, longitude_i, category_i, name_i \rangle$, where id_i is the unique identifier of the POI, $latitude_i$ and $longitude_i$ represent the geographical position of the POI, $category_i$ reflects the category of the specific place (*e.g.*, cinema, bar, etc.) and $name_i$ is the name of the POI (*e.g.*, Starbucks).

Users. Each $user_u \in U$ that is a member of a LBSN can visit POIs and announce his/her spatio-temporal presence there with a "check-in". Whenever a $user_u$ "checks-in" at POI i we record a tuple $c_{u,i} : \langle user_u, POI_i, timestamp \rangle$ that encapsulates the identifiers of the POI and the user as well as the unix timestamp of the "check-in". Finally, we define the variable $f_{u,i}$ that represents the amount of check-ins that the $user_u$ has reported at POI i, this essentially reflects the user preference for each individual POI.

We divide check-ins into groups based on a specific **Context**. The Context can be expressed in many forms and allows us to capture user behaviors related to this context. For instance, considering seasonality as a Context allows us to provide fewer recommendations for coffee places at night compared to morning hours. Seasonality can also consider the day of the week (*e.g.*, weekday, weekend), seasons of the year (*e.g.*, Christmas, summer) or time durations within a day (*e.g.*, morning, noon, night) that users check-in at specific POIs. Another interesting context type is the category transaction. Assume that POI p_1 belongs to category c_1 and POI p_2 belongs to category c_2. We might infer that it is likely that a user will visit POI p_2 after p_1 but never the opposite. An example of this scenario in real life, is that a user may visit a pub after he has visited the gym, but the opposite is very unlikely to happen. In this paper we introduce a model that takes advantage of such contextual information in order to provide more accurate recommendations to users.

III. Methodology

Our methodology consists of two parts: (a) an Initialization phase which extracts the context information from check-ins and initializes a tensor structure, and

(b) a Tensor Factorization phase where we employ a state of the art Stochastic Gradient Descent algorithm to calculate the latent factors for Users, POIs and Context, and then we reconstruct a new tensor which contains the recommendations for each user.

A. Tensor Factorization

Our model exploits Tensor Factorization [8], a generalization of Matrix Factorization, that provides a flexible and generic integration of contextual information.

In the following we first describe the working of our Tensor Factorization approach. We present the loss function (*i.e.*, the objective function) that we use in our model and how we tune the regularization terms to optimize the POIs to recommend, and then we provide our proposed algorithm and can exploit different context types in our model. Finally, we present how we extract the POIs to recommend to the user.

1) Stochastic Gradient Descent algorithm: Our Tensor Factorization approach works as follows. First, we provide our loss function and we add the regularization terms to minimize the objective function. Most of the approaches proposed for solving this optimization problem are based on two methods, Alternating Least Squares[5] and Gradient Descent Methods[7]. The decomposition model we use is based on High Order Singular Value Decomposition (HOSVD). HOSVD introduced by Lathauwer et al. [10] and extended Singular Value Decomposition (SVD) on multi-dimensional arrays. This model includes three latent factor matrices and one core tensor. $U \in \mathbb{R}^{i \times n_U}$,$P \in \mathbb{R}^{j \times n_K}$ and $C \in \mathbb{R}^{k \times n_C}$ and $S \in \mathbb{R}^{k_U \times k_P \times k_K}$.

Although HOSVD works well with dense matrices, it cannot handle missing information. So, further work needs to be done to HOSVD in order to be compatible with Collaborative Filtering. In the next sections we present an algorithm based on the idea of HOSVD that is capable of working well with sparse matrices and missing information.

2) Loss Function: We define a loss function that calculates the distance between approximate and known values.

We use the squared error loss function, so for the case of Tensor Factorization, the objective function is the following:

$$\min_{U,P,C,S} \frac{1}{2} \sum_{i,j,k} R_{i,j,k} (T_{i,j,k} - F_{i,j,k})^2 \qquad (1)$$

where R is the matrix which contains binary rates of each checkin, and $\|\bullet\|_F^2$ is the square root of the sub of the squared values in matrices (also known as the Frobenius norm) and each element in matrix F is given by:

$$F_{i,j,k} = S \times_U U_{(i,\cdot)} \times_P P_{(j,\cdot)} \times_C C_{(k,\cdot)} \tag{2}$$

Any tensor can be represented as a 2D array, we call this procedure *unfold*. In the above, $S\times_U, S\times_P, S\times_C$ represent the corresponding unfolding for core tensor S, in order to let us do multiplications with matrices U, P, C. A tensor can be unfolded into any of its dimensions. The unfoldings of a tensor $T \in \mathbb{R}^{I\times J\times K}$ are shown in Figure 1. Tensor unfolding allows us to make matrix-tensor manipulations. This way, Tensor S^{D_1,D_2,D_3} and three matrices U^{I,D_1}, P^{J,D_2} and C^{K,D_3} the notation $S \times_U U$, represent that we unfold tensor S so it can be multiplied with matrix U.

3) Tensor decomposition: Given the T tensor, one possible approach to POI recommendation is to apply tensor factorization with respect to it. This involves mapping users, POIs and context into a joint latent space model. The model includes three latent factor matrices and one core tensor. $U \in \mathbb{R}^{i\times n_U}$,$P \in \mathbb{R}^{j\times n_K}$ and $C \in \mathbb{R}^{k\times n_C}$ and $S \in \mathbb{R}^{k_U\times k_P\times k_K}$. Each of the n_U, n_P, n_C represent the dimension of the corresponding latent space, where $n << \min(I, J, K)$. This way a user's preference for a POI on a contextual category is modeled as the inner product between these three matrices and the core tensor in that latent space. The mapping is achieved by approximating the frequency matrix by solving the optimization problem of Equation 1.

4) Regularization: If we try to solve equation 1 we will face the well known problem of over-fitting [13], where our model will strive to minimize the function as complexity grows. For each of the matrices U, P, C and tensor S we will add a regularization term based on the Frobenius Norm. Thus, our objective function will be:

$$\begin{aligned}\min_{U,P,C} (\|R \odot (T - F)\|_1^2) + \gamma(\|U\|_F^2 \\ + \|P\|_F^2 + \|C\|_F^2)\|_F^2) + \sigma \|S\|_F^2\end{aligned} \tag{3}$$

The variables γ and σ are the regularization terms used to avoid the over-fitting of our model and we can find them by applying cross-validation to our dataset. We would like to emphasize, that, regularization terms are a core factor to our model's predictions, since if they are initialized with small values (*i.e.*, 10^{-4}) our objective function will converge to 0, and this means that our model will be trained to converge to the known ratings and may diverge from the unknown ones (*i.e.*, over-fitting) that leads to poor predictions. On the other hand, assigning a large value to our regularization terms will lead to very generic predictions which will lead us again to poor predictions.

5) Optimization using stochastic gradient descent: We perform the optimization using the stochastic gradient in the factors $U_{i\bullet}$,$P_{j\bullet}$, $C_{k\bullet}$ and the core tensor S for a given rating from tensor $R_{i,j,k}$ simultaneously, as proposed in [7]. Due to the large size of the datasets we consider, it becomes impractical to use batch optimization techniques to solve this factorization. To overcome this problem we use the following online algorithm which performs stochastic gradient descent, on the observed ratings of $Y_{i,j,k}$, for the factors $U_{i,\bullet}$, $P_{j,\bullet}$, $C_{k,\bullet}$ and core tensor S simultaneously. Thus, we find the derivatives of the objective function with respect to U, P, C and S. We show the derivatives of U, S below (calculating the derivatives of P and C is similar to U).

$$\frac{dR}{d_{U_{i,\bullet}}} = R_{i,j,k}(F_{i,j,k} - T_{i,j,k})S \times_P P_{j\bullet} \times_c C_{k\bullet} + \gamma U_{i\bullet} \tag{4}$$

$$\frac{dS}{d_{S_{i,\bullet}}} = R_{i,j,k}(F_{i,j,k} - T_{i,j,k})U_{i\bullet}^{-1} \otimes P_{j\bullet} \otimes C_{k\bullet} + \sigma S \tag{5}$$

Note, that, $\otimes$ represents the *Kronecker product* in two matrices. Given matrices $A \in \mathbb{R}^{I,J}$ and $B \in \mathbb{R}^{K,L}$ the Kronecker product is defined as $A \otimes B$, which leads to a matrix with size $(IK) \times (JL)$.

The Kronecker product is used in order to reconstruct the core tensor S using matrices $Utemp$, $Ptemp$ and $Ctemp$ where $Utemp = U_{i\bullet}^{-1}$, $Ptemp = P_{j\bullet}$ and $Ctemp = C_{k\bullet}$. After finding the Kronecker product for the three latent factor matrices, we can fold the matrix again and the result matrix will lead us to the core tensor S.

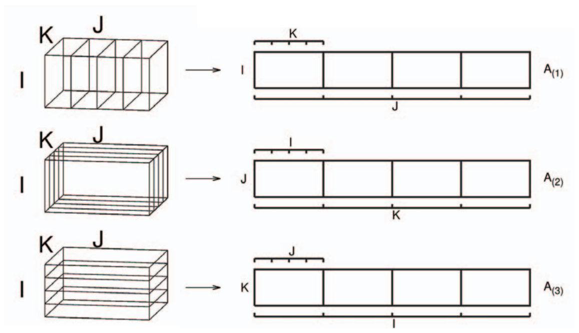


Fig. 1: 3D Tensor Unfoldings.

The Tensor Factorization algorithm we used is summarized in Algorithm 1. As can be seen, each i, j, k record is updated independently, so this algorithm can be easily parallelized. In every iteration U, P and C are set to $tempU$, $tempP$ and $tempC$ respectively. This happened to ensure that in every step U, P, C and S will not be affected by the update of U, P, C, and S (the order of the updates does not matter). α is the learning rate we use in our Stochastic gradient descent algorithm. Its initial value and update parameter $step$ are considered after applying cross-validation to our dataset. The procedure repeats until our objective converges. This varies with the size of the observed values in the dataset we use and the size of the latent space. Finally, using the output tables U, P, C and tensor S, we can calculate every element of the recommendation matrix $\tilde{R}$ using Equation 2.

6) Shuffling: Updating the latent factors using Stochastic Gradient Descent algorithms need many adjustments to coverage quickly and provide accurate results. We already mentioned how the regularization terms and the learning rate affect the Tensor Factorization approach. Based on the observation of [4] we identify another important factor that can highly

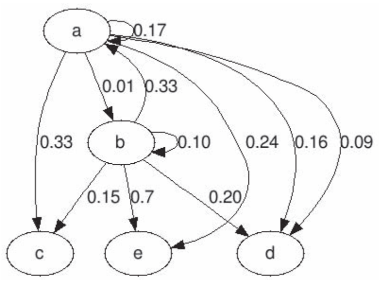


Fig. 2: A category transition graph

improve running times and achieve better recommendations. To achieve that we shuffle the known ratings before every iteration.

The ratings in our tensor are ordered in each one of its dimensions (*i.e.*, users in I axis, POIs in J and context in K). To achieve this, we create a list of the positions of the observed values of our initial tensor Y. Upon every iteration we shuffle that list, and then we read the observed values from the tensor based on the list's order. This enables us to ensure that latent factors will not be updated in any user/POI/context sequential pattern.

Algorithm 1 Tensor factorization algorithm

```
Input: $T, \gamma, \sigma, n_U, n_P, n_C, epochs$
Initialize $U \in \mathbb{R}^{i \times n_U}$, $P \in \mathbb{R}^{j \times n_K}$, $C \in \mathbb{R}^{k \times n_C}$, $S \in \mathbb{R}^{n_U \times n_P \times n_K}$
$R^{i,j,k}$ binary of $T$
for $e = 0$ to $epochs$ do
    for all $(i, j, k)$ where $R_{i,j,k} > 0$ do
        $\alpha = t$ and $t = t + step$
        $\tilde{R}_{i,j,k} = S \times_U U_{(i,\cdot)} \times_P P_{(j,\cdot)} \times_C C_{(k,\cdot)}$
        $tempU = U_{i,\bullet}$
        $tempC = C_{c,\bullet}$
        $U_{i,\bullet} = tempU - \alpha\gamma tempU - \alpha(R_{i,j,k} \times (T_{i,j,k} - F_{i,j,k})S \times_P P_{j\bullet} \times_c C_{k\bullet})$
        $p_{j,\bullet} = tempP - \alpha\gamma tempP - \alpha(R_{i,j,k} \times (T_{i,j,k} - F_{i,j,k})S \times_U U_{j\bullet} \times_c C_{k\bullet})$
        $C_{k,\bullet} = tempC - \alpha\gamma tempC - \alpha(R_{i,j,k} \times (T_{i,j,k} - F_{i,j,k})S \times_U U_{i\bullet} \times_P P_{j\bullet})$
        $S = S - \alpha\sigma S - \alpha(R_{i,j,k}(T_{i,j,k} - F_{i,j,k}) fold_U(tempU \otimes tempP^T \otimes tempC^T))$
Output: U, P, C, S
```

B. Context-Aware Point Of Interest Recommendation

In this section we examine how *category-based context* can be used for Point Of Interest recommendation. Note, that, it is very important to initialize tensors in a way that we will be able to take advantage of the multidimensional structures and the type of contexts that our dataset provides to us.

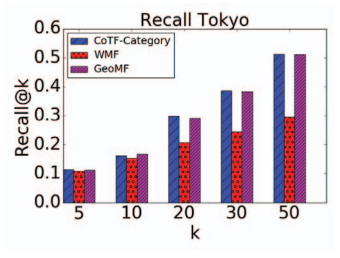
In real life users tend to have specific patterns when visiting POIs. For example a user u usually check-ins at his work at the morning, in a restaurant at noon and at a pub or at a cinema at night. On the other hand, it is not so common for someone to visit the gym after he visited a pub. More formally, an example of this is shown in Figure 2. Assume categories A, B,C, D and E. We also define POIs $a \in A$, $b \in B$, $c \in C$, $d \in D$ and $e \in E$. The path $a \rightarrow b \rightarrow c$ is unlikely to take place, on the other hand $a \rightarrow c$ is more likely to happen. We also found that a lot of times users tend to choose a POI of the same category for their next check-in (e.g. $a \rightarrow a$ has a probability of 17% to happen). These patterns in category transition have been studied by Liu *et al.* in [12] and the solution he proposed was based on pre-filtering techniques to datasets before applying Matrix Factorization to User/POI matrices. The main problem of the existing approaches is that they brought extra complexity to the models. The other important drawback is that they require a lot of terms to be found using cross validation, which is a time consuming process. In our approach we will use a tensor $T \in \mathbb{R}^{I,J,K}$ where I is the number of users, J is the number of POIs and finally K is the number of categories for the POIs. In this case, k represents the previous category of the current user's check-in. This way a check-in of user i to a POI j within context k, means that before visiting POI j, user i has been to a POI where its category was k. So, if a user is at a POI of category k and he would like to receive recommendations, we will use user's i and category's k feature vectors to calculate the rating for each POI. This way we fully utilize the 'Last Category' context to our model. Our experimental results clearly show the improvement of the results using this technique.

IV. Experimental Evaluation

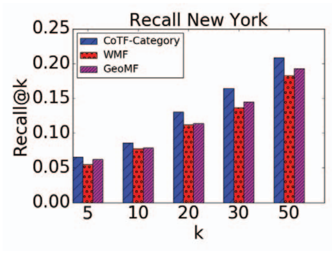
Dataset Description and Compared Methods. We have used two large-scale location based social network datasets to evaluate the working and benefits of our approach. Our datasets were crawled from Foursquare and include (i) a New York Foursquare dataset that took place in New York, USA between 12 April 2012 and 16 February 2013, comprising 1083 users with 227,428 check-ins on 38,333 POIs, and (ii) a Tokyo, Japan Foursquare dataset between 4 April 2012 and 16 February 2013, comprising 2293 users with 573,703 check-ins on 61,858 POIs. The Foursquare datasets were also used by Yang *et. al.* [17]. These datasets are appropriate for the evaluation of our approach as they contain POIs with unique IDs. Using these IDs, we were to obtain POIs' categories using the Foursquare API. The average user/POI slice density is 2.19×10^{-3}. We filtered out users who have been to fewer than 10 distinct locations and POIs which have been visited by fewer than two users; this resulted into 1083 users and 8009 POIs left in New York and 2293 users and 14508 POIs left in the Tokyo dataset respectively.

For each user, we select randomly 20% of her visiting locations as ground truth for testing, while the remaining locations for the user consist of the training dataset for learning the parameters of our model. Then, for each user, all locations from the learned model are ranked, and the ones included in the training set are finally excluded. The learned model is then assessed by its capacity to find the ground truth locations for each user among the ranked locations. To measure this we used Recall@k and Precision@k, in the top-k POI recommendation. More formally we define $K_u(k)$ as the top-k recommended POIs for user u. V_u are the visited locations of user u and where M is the total number of users. Hence, we define Recall@k and Precision@k as follows:

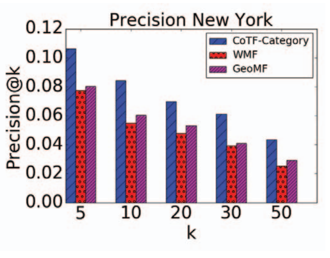
$$Recall@k = \frac{1}{M} \sum_{u=1}^{M} \frac{|K_u(k) \cap V_u|}{|V_u|} \tag{6}$$



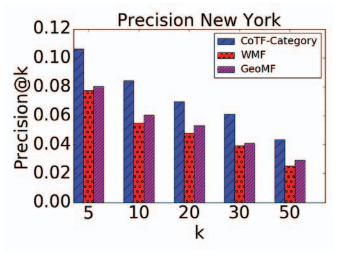
(a)



(b)



(c)



(d)

Fig. 3: Recall and Precision for the Tokyo (a),(c) and New York (b),(d) datasets.

$$Precision@k = \frac{1}{M}\sum_{u=1}^{M}\frac{|K_u(k) \cap V_u|}{|k|} \tag{7}$$

All the results we present are averaged over 10 independent runs. We compare our results with two state-of-the-art methods on implicit feedback and POI Recommendation:

- The first one is the **Weighted Matrix Factorization Method** presented by Hu *et. al.* [6] which is known to cope well with the implicit feedback. The objective function of the above method is given by:
$$\min_{U,P}(\left\|W \odot (R - UP^T)\right\|_F^2) + \gamma(\|U\|_F^2 + \|P\|_F^2) \tag{8}$$
where W is the matrix a rating matrix, is a weighted version of matrix C which is the initial matrix that contains the user-POI visit frequencies and R is a binary matrix that indicates if a user u has visited POI p.
- The second one is the **GeoMF algorithm** presented by Lian et. al. [11]. GeoMF algorithm augments the WMF method, by adding a concept called activity areas. Activity areas are trying to isolate undiscovered but interesting places for the user places, from the uninteresting ones:
$$\begin{aligned}\min_{U,P,X}&(\left\|W \odot (R - UF^T - XY^T)\right\|_F^2)\\ &+ \gamma(\|U\|_F^2 + \|P\|_F^2) + \lambda\left\|X\right\|_1)\end{aligned} \tag{9}$$
where $X \in \mathbb{R}^{U \times L}$ and $Y \in \mathbb{R}^{P \times L}$. Y indicates the POIs' influence areas, *i.e.,*, the geographical influence area of each POI which decays as we draw away from the POI following a normal distribution. Similarly, we denote the users' activity areas as X, that indicate the visits of a POI in region grid l.

For the latent space we chose n=40 for our model and n=40 for GeoMF and WMF, which were large enough to fit the need of the dataset we used. All of the above models were implemented in Java using the high performance ND4J [15] library. We run all the experiments on a i5-6600k processor with 16GB RAM.

Experimental Results. Figures 3a, 3b, 3c and 3d present the results of the comparison of our method with the WTF and GeoMF techniques. We measured Recall and Precision at 5, 10, 20, 30, 50.

As shown in figures 3a, 3b, 3c and 3d our approach outperforms WMF in both datasets for all values of N. In the case of the GeoMF, our approach has similar or better performance than GeoMF. More specifically in the New York dataset, Recall was improved from 6% $@k = 5$ to 8% $@k = 50$, while Precision was further improved, from 32% $@k = 5$ to 48% $@k = 50$. In the Tokyo dataset, we had 2% $@k = 5$ improvement on Recall and similar performance on higher values of k, while precision was improved in all cases, from 21% $@k = 5$ to 15% $@k = 50$.

Another important improvement of using Tensor Factorization instead of augmented Matrix Factorization models is the execution times. GeoMF uses Alternating Least Squares based algorithm (an extension of WMF) to calculate users' and POIs' latent factors and then a Batch Gradient Descent method to update user activities areas. This process goes iteratively until coverage. Assuming a grid with a typical cell size is 1km, then New York consists of 50 vertical and 47 horizontal cells. GeoMF is also depends its functionality on matrices W, X and Y. All these are extremely sparse matrices. To support such big matrices to fit PC memory, we had to implement a sparse array structure. This lead us to the drawback of even longer running times due to indexing needed to support such structures. The running times of the training phase of all methods, for New York dataset, are summarized in table I.

Model	Training Time (in seconds)
CoTF	2800
GeoMF	422332
WMF	6750

TABLE I: Training Time Comparison.

It is remarkable, that, compared to GeoMF, the relative times of the training phase are much smaller. This is due to the fact that, GeoMF, uses a Batch Gradient Decent method to calculate User's Activity Areas, which is computational expensive, and has to be re-calculated for every outer iteration. Finally we have to note that, in order to get right results, for the New York dataset regularization terms have to be set at $k_u = k_p = 0.01$ and $k_c = 0.001$ and for the Tokyo data set $k_u = k_p = 0.001$ and $k_c = 0.001$

V. Related Work

With the rapid grown of LBSN, several approaches for POI recommendation have emerged in recent years. The authors of [19] propose a graph-based approach over time-aware POI recommendation. However, all the approaches mentioned above consider only one dimension *spatial or temporal*, while CoTF can handle different types of context. Authors in [6] introduced a Weighted version of Matrix Factorization (WMF), where users and POIs are mapped into a joint latent space by approximating a user-POI binary rating matrix in a weighted way. Despite of that, this method can handle implicit feedback, is based on Matrix Factorization and thus, does not take context into account. Lian *et al.* in [11] augmented users' and POIs' latent factors of the factorization model with non-negative activity area vectors of users and non-negative influence area vectors of POIs to improve the accuracy of POI recommendations. Although this method can work with implicit feedback and take POI location into account to cope with any other kind of context we need to modify the model itself. Thus, this method using a batch gradient descent method to update the activity areas, which is a very slow and memory consuming method. Approaches that incorporate context have been recently proposed in order to improve the accuracy in POI recommendation. In [12] researchers tried to find the category transition. In order to achieve this, they apply pre-filtering techniques to prepare the data for factorization. After the factorization they apply post-filtering techniques to filter the results using the location information. However, such an approach reflects a significant overhead compared to our Tensor-based approach. Another context-aware Matrix Factorization technique is a reduction based approach, which is based on OLAP [2] and extends classical Collaborative Filtering methods by adding contextual information to the representation of users and items. Baltrunas *et al.* [3] propose a technique that splits items with extreme range in the ratings. Both of the above two methods are pre-filtering techniques that exploit Matrix Factorization, they are based on explicit feedback and they lead to sparser matrices. Our approach was to include all the above ideas to a more general and flexible method. To achieve this we used a stochastic gradient descent method. Karatzoglou *et al.* [7] introduced a Tensor Factorization method capable of handing different kinds of context, without the need of pre-filter or post-filtering techniques. His approach outperformed the above two techniques in terms of Mean Average Error, on movie and food recommendation. Hidasi *et al.* [5] proposed an TF method based on Alternating Least Squares capable of coping with implicit feedback, and have evaluated its performance in the case of music or movie recommendations. The main drawback of this approach is the intermediate data explosion problem which means the amount of intermediate data becomes very large, although the input and the output are not too large.

VI. Conclusions

With the rapid adoption of Location Based Social networks, the concept of taking advantage of the big amount of available data and making personalized recommendations to their users, has become an important part of these networks. In this paper we have proposed Context-Aware Point of Interest Recommendation (CoTF) system, that exploits Tensor Factorization to construct the recommendations tensor, taking advantage of the contextual information embedded in check-ins, such as the POI category. Our experimental evaluation illustrates that CoTF efficiently produces personalized recommendations to users while significantly reducing the training time compared to state of the art approaches.

Acknowledgments

This research has been financed by the European Union through the FP7 ERC IDEAS 308019 NGHCS project.

References

[1] Foursquare. https://foursquare.com/.

[2] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, Jan. 2005.

[3] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, pages 245–248, New York City, NY, October 2009.

[4] L. Bottou. Stochastic gradient tricks. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700), pages 430–445. Springer, 2012.

[5] B. Hidasi and D. Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. *CoRR*, abs/1204.1259, 2012.

[6] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, Pisa, Italy, December 2008.

[7] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, Barcelona, Spain, September 2010.

[8] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[10] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, Mar. 2000.

[11] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *SIGKDD*, pages 831–840, New York, NY, August 2014. ACM.

[12] X. Liu, Y. Liu, K. Aberer, and C. Miao. Personalized point-of-interest recommendation by mining users' preference transition. In *CIKM*, pages 733–738, San Fransisco, CA, October 2013.

[13] A. Y. Ng. Preventing "overfitting" of cross-validation data. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 245–253, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[14] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, Pisa, Italy, December 2008.

[15] N. D. Team. Nd4j: N-dimensional arrays and scientific computing for the jvm, 2016. [Online; accessed 19-April-2016].

[16] Y. Wang, J. Qu, J. Liu, J. Chen, and Y. Huang. What to tag your microblog: Hashtag recommendation based on topic analysis and collaborative filtering. In *Web Technologies and Applications*, pages 610–618. Springer, 2014.

[17] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.

[18] W.-Y. M. Yu Zheng, Xing Xie. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data(base) Engineering Bulletin*, June 2010.

[19] Q. Yuan, G. Cong, and A. Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 659–668, New York, NY, USA, 2014. ACM.

[20] Y. Zheng. Location-based social networks: Users. In *Computing with Spatial Trajectories*, pages 243–276. Springer, 2011.

[21] Y. Zheng. Tutorial on location-based social networks. In *WWW*, volume 12, Lyon, France, April 2012.