

Who, What, When, and Where: Multi-Dimensional Collaborative Recommendations Using Tensor Factorization on Sparse User-Generated Data

Preeti Bhargava^{1*}, Thomas Phan², Jiayu Zhou², Juhan Lee²

¹Department of Computer Science, University of Maryland, College Park

²Samsung Research America – Silicon Valley

prbharga@cs.umd.edu, {thomas.phan,jiayu.zhou,juhan.l}@samsung.com

ABSTRACT

Location recommendations have become very popular with the advent of smartphones via which useful information and services can be delivered to the users instantly. In addition to location recommendations, users would also benefit from recommendations for activities to participate in at the location along with a suitable time of participation. Thus, recommender systems that generate collaborative recommendations involving multiple dimensions such as location, time, and activities would enhance the overall experience of users and provide them with the most helpful recommendations. In these systems, the relationship among the various dimensions is modeled by higher-order matrices called tensors which are then solved by tensor factorization. Moreover, sparsity is a major concern for user generated data. In this paper, we present a system and an approach for performing multi-dimensional collaborative recommendations for Who (User), What (Activity), When (Time) and Where (Location), using tensor factorization on sparse user generated data. We formulate an objective function which simultaneously factorizes coupled tensors and matrices constructed from heterogeneous data sources. We evaluate our system and approach on large-scale real world data sets consisting of 588,000 Flickr photos collected from three major cities in the USA. We compare our approach with several state of the art baselines and demonstrate that it outperforms all of them.

Categories and Subject Descriptors

C.3.2 [Special-Purpose and Application-based Systems] : Real-time and embedded systems

General Terms

Algorithms; Design; Experimentation; Human Factors; Performance

Keywords

Recommender systems; Multi-dimensional recommendations; Collaborative recommendations; Tensor factorization

1. INTRODUCTION

Today's smartphones come equipped with a multitude of sensors such as GPS and an increasing range of computational, storage and communication capabilities. This has enabled several smartphone based applications to emerge across a wide variety of areas such as location based services, social networking and healthcare. Moreover, through these devices, information can be delivered to users instantly. However, with the abundance of information available to us, it is important that only relevant and personalized information is presented to the users. As a result, recommender systems, that suggest items of interest to people based on their context and preferences, have become increasingly ubiquitous and popular.

*This work was conducted while the first author was an intern at Samsung Research America – Silicon Valley.

Since the GPS embedded in smartphones can be used to accurately localize a user, *location* has become the basis of many recommender systems. These systems recommend interesting places or landmarks for visiting to users. However, in location recommendations, contextual information such as *time* often plays an important role as certain places are open only between fixed hours or can host certain activities at fixed times of a day or on fixed days in a week. In addition, users who want suggestions or recommendations for places to visit would also benefit from recommendations for *activities* to participate in at the location along with a suitable time of participation. Here, activities refer to human lifestyle and recreational activities such as shopping, dining, surfing etc. Users may even have specific *preferences* for activities such as preferring outdoor rather than indoor activities. For instance, at Pier 39 in San Francisco, a list of all possible things that a tourist can do includes shopping, eating seafood at the various restaurants, riding the Venetian carousel, watching the sea lions at the next door dock or a getting a view of the Fourth of July fireworks display. However, all these activities happen at specific locations in Pier 39 and either in specific months, on specific days or at specific hours. Thus, recommender systems that generate collaborative recommendations involving multiple dimensions such as location, time, activities, and user preferences etc., would enhance the overall experience of users and provide them with the most helpful recommendations.

As a simplest formulation, recommender systems model preferences of *users* for *items* in the form of a *Utility Matrix* where rows represent users, columns represent items and the values represent the users' ratings or preferences for those items on a scale of say, 1 - 5. The goal of the recommender system is then to impute the missing values based on observed values in the matrix [28, 38] using a standard approach such as Collaborative Filtering. User-based collaborative filtering determines a subset of users most similar to the current user, and predicts the missing ratings based on a weighted combination of the ratings provided by them [28]. Similarly, item-based collaborative filtering [25] focuses on predicting the missing ratings for items based on a weighted combination of the ratings given by the current user to similar items. Similarity between each pair of items is determined by the similarity of the ratings of those items provided by the users who have rated both items. Once the matrix has been completed, a user can be given a recommendation list of ranked items ordered by descending predicted ratings.

An alternative powerful approach, that has been used with positive results in recommender systems, is Latent factor models [23]. It emerged from the research that was fueled by The Netflix prize [7]. Unlike neighborhood based methods, latent factor models assume that the similarity between users and items is simultaneously induced by some hidden lower-dimensional structure in the data. Some of the most successful realizations of latent factor models are based on matrix factorization, where users and items are simul-

taneously represented as unknown feature vectors (column vectors) along k latent dimensions. These methods have become prominent in recent years because they combine scalability with high predictive accuracy. In addition, they offer more flexibility for modeling various practical scenarios where the data is very sparse.

In case of recommendations along a single dimension, the 2 dimensional user \times item matrix factorization model can be applied successfully. For instance, location recommendations can be modeled as a user \times location matrix and can be solved using standard approaches to matrix factorization. However, to model collaborative multi-dimensional recommendations involving location, activity, time and other contextual information, the relationship among the various dimensions is represented by higher-order matrices called *tensors*. To address them, standard matrix factorization approaches need to be generalized to tensor factorization.

Another major challenge in recommender systems is that, in practice, most users provide interest ratings for only a subset of the recommended items. For instance, a user will provide interest ratings for only a subset of all the possible locations. Moreover, as the number of dimensions increase, the sparsity of data increases and it becomes a major concern for recommender systems that generate multi-dimensional recommendations from user generated real world datasets.

In this paper, we address these challenges and present a system and an approach for performing multi-dimensional collaborative recommendations for Who (User), What (Activity), When (Time) and Where (Location), using tensor factorization on sparse user generated data. Our contributions in this paper are:

- We address n -dimensional collaborative recommendations (where $n \geq 4$ and includes user, location, activity, time and any other contextual information) by fusing data from multiple sources such as Flickr (a photo sharing website), Foursquare (a location based social network), Yelp (a crowd-sourced reviews based website) and Viator (a travel website).
- We formulate an objective function, that simultaneously factorizes coupled sparse tensors and matrices constructed from heterogeneous data sources, and minimize it using a gradient descent based algorithm.
- We evaluate our system and approach on large-scale real world data sets consisting of 588,000 Flickr photos collected from three major cities in the USA – the San Francisco Bay Area, Las Vegas and Chicago. The tensors constructed from these datasets are 99.999% sparse.
- We compare our approach with several state-of-the-art baselines and demonstrate that it outperforms all of them.
- Our approach also demonstrates an improvement in runtime without the need for sacrificing performance.

The rest of the paper is organized as follows: Section 2 reviews related work in location and activity recommendations and tensor factorizations applications to recommender systems. In Sections 3 and 4, we describe our datasets and the information inferred from them. We explain our approach in Section 5 and present its evaluation and comparison with baselines in Section 6. Finally, we conclude and outline future work in Section 7.

2. RELATED WORK

Since our work involves multi-dimensional recommendations for location, activity and time using tensor factorization on extremely sparse user generated data, we have categorized the related work into different sections. The first section covers existing literature in location and activity recommendations while the second covers existing work in recommender systems using tensor factorization. We differentiate our work from them and identify their shortcomings.

2.1 Location and Activity recommendations

Most of the existing literature has focused on recommendations

along one dimension (for instance, location). A few researchers have explored collaborative location and activity recommendations. To the best of our knowledge, none of the works have attempted collaborative location, activity and time recommendations.

2.1.1 Photos as a data source

One of our main data sources for user locations comes from the Flickr¹ website for sharing photos. Many of the photos uploaded by users are geotagged, thereby providing a wealth of geospatial data. These photos have been used for many purposes, such as finding Points-of-Interest (POI) clusters [39], identifying the location of photos from visual, textual, and temporal features [11], determining when tourism is in season [14], and creating routes that are pleasing to the user [33].

2.1.2 Location/Landmark/Venue recommendations

Previous researchers have investigated the problem of making recommendations for geolocations that may be interesting to the user [5, 9, 26, 35]. Such approaches use large bodies of collected geospatial data along with user preferences and then apply low-dimensional recommendation algorithms.

2.1.3 Activity Recommendations

Even though users seek recommendations on what activities they can engage in when they visit a place or at a given time, the area of activity recommendation has not been researched extensively. A majority of the current research has focused on diurnal activity recognition from smartphone sensory data [29, 27] or from location [24]. Belotti et al. [6] explored the idea of serendipitous activity based discovery of venues and activities via the Magitti Mobile Leisure Guide. Magitti first infers the ongoing and future activities, then predicts what information will be most useful to the user during the predicted activity and further personalizes it. Ducheneaut et al. [13] experiment with several models such as Collaborative Filtering, preference based, distance based, and a weighted combination of these to provide activity recommendations via Magitti.

2.1.4 Collaborative Location and Activity recommendations

Co-occurring location and activity collaborative recommendations were proposed by Zheng et al. in [41] where they addressed location recommendations given an activity, and activity recommendations given a location. They used Collective Matrix Factorization (CMF) [36] to complete a sparsely populated 2 dimensional Location \times Activity matrix and evaluated their approach on 162 users with recommendations for 5 activities. CMF takes advantage of correlations and sharing of information between data sets from multiple sources and simultaneously factorizes coupled matrices. It is shown to have achieved higher prediction accuracy than individual matrix factorization. Sattari et al. [34] used the same dataset as Zheng et al. in [41] but employed Singular Value Decomposition (SVD) in place of CMF to complete the Location Activity matrix. Since SVD requires the matrix that needs to be decomposed to be fully populated, they padded it with zero values. They demonstrated an improvement in performance over [41].

Zheng et al. [40] modeled user, location and activity data as a 3 dimensional tensor and applied a regularized tensor and matrix factorization approach for location and activity recommendations. They formulated a CANDECOMP/PARAFAC (CP) [17, 10] decomposition style objective function and minimized it using gradient descent. This decomposition factorizes a tensor into a linear combination of component rank-one tensors. They evaluated their approach on a dataset of 164 users, 168 locations and 5 activities.

Our work stands out from the existing literature in several ways:

¹<https://www.flickr.com/>

Data Source	Type of Data	Dimensions extracted	Volume of raw data extracted
Flickr	Geotagged and Timestamped Photos	User, Location, Activity, Time	588,000 photos with 9 million words of text
Foursquare	Location and POI database	Location, Activity, Venue	POI or Venue information for 274,000 locations
Yelp	Business and service reviews	Activity	1060 service categories
Viator	Things to do in tourist spots	Activity	60 things to do categories

Table 1: Data fusion from heterogeneous data sources

Region	Photos	Users	Land Area (km ²)
San Francisco Bay Area	280,045	7895	6053
Las Vegas	23,350	578	1818
Chicago	284,751	2423	1412

Table 2: Flickr.com dataset characteristics

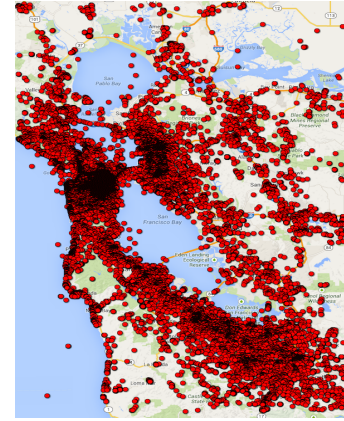
- We use a large user generated dataset which involves several thousand users and locations, and over a hundred popular lifestyle, recreational and tourist activities.
- We incorporate the dimension of context such as time in addition to location and activities.
- As opposed to a manual approach for activity inference from user comments that was employed in [41, 40], we propose an automated and unsupervised Natural Language Processing (NLP) based algorithm (Section 4.2) which is more robust and scalable to large real-world datasets.
- Moreover, a major limitation of using CP decomposition for tensors is that it is not suitable for very sparse tensors and demonstrates a sharp increase in error especially if more than 80% of the data is missing [2, 3]. Sparsity is a non trivial issue for us as our multi-dimensional data is 99.999% sparse. Hence, we propose our approach to solve the problem.

2.2 Other applications of Tensor Factorization to Recommender Systems

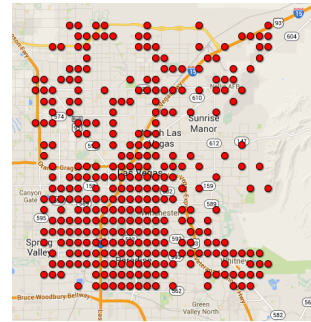
Symeonidis et al. [37] and Nanopoulos [31] applied Higher Order Singular Value Decomposition (HOSVD) [12] to a 3rd order tensor which represents users, items and tags in social tagging systems such as Last.fm and Bibsonomy. Karatzoglou et al. [20] address multi-dimensional recommendations by incorporating contextual information to model a User-Item-Context tensor. They utilize a sparse HOSVD style method that decomposes a D dimensional sparse tensor into D matrices and a D dimensional tensor. HOSVD is a generalization of the matrix SVD to a tensor. It assumes a dense tensor and is not suitable for very sparsely populated tensors. Moreover, unlike SVD, HOSVD may not provide the best low rank approximation of a tensor [22].

Hidasi and Domonkos [19] apply an Alternating Least Squares (ALS) [21, 22] based tensor factorization approach for context-aware recommendations. ALS consists of three steps, each one being a conditional update of one of the factor matrices, given the others. However, it suffers from several drawbacks: It has poor convergence in the presence of missing data [8] and it does not scale to large-scale data sets [2].

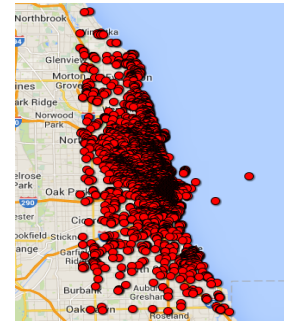
To address these limitations and perform multi-dimensional recommendations on large-scale and sparse user-generated datasets, we formulate our recommendation model via the Coupled Matrix and Tensor Factorization (CMTF) framework [1, 3]. CMTF is an approach similar to CMF and proposes joint analysis of a matrix and an Nth-order tensor with a common mode or dimension, where the tensor is factorized using an R-component CP model and the matrix is factorized by extracting R factors using matrix factorization. CMTF is shown to have achieved better performance than standard CP decomposition especially if more than 80% of the data is missing [2, 3]. A variant of the CMTF approach [1, 2] performs the joint analysis of the tensor and matrix by ignoring the missing entries and fitting the tensor and/or the matrix model to the known



(a) San Francisco Bay Area



(b) Las Vegas



(c) Chicago

Figure 1: Geographical distribution of photos from San Francisco Bay Area, Las Vegas and Chicago core regions.

data entries only. This approach has been shown to easily scale to handle data with missing entries even for very large data sets with up to 99% missing entries [2].

To this end, we model our multi-dimensional recommendation problem as a joint analysis of a sparsely populated tensor with several matrices which share one or more common modes with the tensor. These tensors and matrices have been constructed by fusing data for the various dimensions (users, locations, activities and time) from multiple data sources. This is a challenging task since data sets are often incomplete and heterogeneous. We then factorize these tensors and matrices simultaneously using gradient descent. As we show later in Section 6.5, our approach outperforms standard CP decomposition, HOSVD and ALS.

3. DATA

Table 1 summarizes the data sources, types of data, extracted dimensions and the volume of raw data from each source. Our primary dataset consists of 588,000 geotagged and timestamped publicly-available photos from Flickr.com, a popular photo-sharing website hosted in the USA. To obtain the data set, we searched specifically for photos taken with smartphones so that we could get the most accurate geospatial traces. Further, these photos spanned

Algorithm 1: Cartographic Sparse Hashing (CASH) algorithm

Input: *latitude* as 64-bit float, *longitude* as 64-bit float, grid resolution *r* in meters

Output: Hash value *hashResult* as 64-bit integer

Trim *latitude* digits past 5th decimal position;

$sigFig \leftarrow 10^5$;

$latitudeInt \leftarrow (int)(latitude \times sigFig)$;

Round down *latitudeInt* to be divisible by *r*;

Repeat with *longitude* to produce *longitudeInt*;

$hashResult \leftarrow shiftHigh(longitudeInt) + latitudeInt$;

return *hashResult*

the time period of September 1, 2009, to September 1, 2013 and were taken from three major metro regions in the USA – the San Francisco Bay Area, Las Vegas and Chicago. Table 2 shows the characteristics of these datasets, while Figure 1 shows the geographical distribution of the photographs over the three regions. To improve legibility of the figure, so that individual photo locations can be discerned, we sampled the number of photos down.

We extracted the photos and their meta-content in JSON format using Flickr’s public REST based API. Each photograph is marked with the user ID of the user who took it, geo-location in latitude and longitude format representing where it was taken, and an epoch timestamp representing when it was taken. We also extracted about 9 million words of user generated text (including title, description, tags or comments) for these photographs. Note that less than 50% of the photographs are marked with any text. In addition, we also obtain POI or Venue information for 274,000 locations in the three geographical regions from Foursquare² (a location based social network). Finally, we obtained information about popular activities from Yelp (a crowd-sourced reviews based website)³ and things to do in tourist spots from Viator⁴ (a travel website). We explain the inference and extraction of the various dimensions from the collected data in Sections 4 and 5.

4. INFERRING VARIOUS DIMENSIONS OF INFORMATION FROM FLICKR PHOTOS

As mentioned earlier, photos in the Flickr datasets have meta-data that includes a user ID, a timestamp, location and photo content. We infer the various dimensions of user, location, activity and time from it.

4.1 Location Hashing

We discretized each location (in latitude and longitude format) into virtual rectangular bins that are formed throughout the geographical coordinate space, where each bin is created through Cartographic Sparse Hashing, our $O(1)$ hash algorithm shown in Algorithm 1. This hashing takes as input: (i) latitude and longitude as 64-bit floats and (ii) a grid resolution size *r* in meters. The function leverages the fact that the latitude and longitude geocoordinates are expressed in decimal degrees, where the precision of the 5th decimal place corresponds roughly to 1 meter. Because this degree of geocoordinate precision was acceptable to us, we trimmed the floating point values to that number of decimal places. The function then outputs a 64-bit integer key, where the longitude and latitude end up in the high and low bits, respectively. The resulting key represents a virtual bin approximately *r* meters per side, although the bin will be elongated north-to-south as one gets further away from the equator due to the Earth’s curvature.

²<https://foursquare.com/>

³<http://www.yelp.com/>

⁴<http://www.viator.com/>

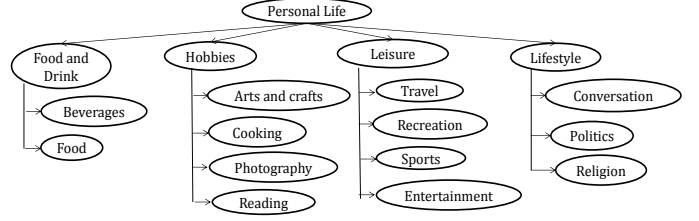


Figure 2: Partial view of the Activity Hierarchy showing all depth 1 nodes and a few depth 2 nodes

We experimented with 4 different location grid sizes - 300m, 500m, 700m and 1000m. These were determined based on the venue density in each grid as well as human walking distance (as people may often prefer to walk or use public transport). If the grid size is too large, recommendations beyond a certain walking distance will not be helpful to the user. On the other hand, if the grid size is too small, there may not be any venue or place to recommend in the grid. We explain the selection of the appropriate size based on performance in Section 6.3.

4.2 Activity Inference

Previous research in activity recognition [24, 27, 29] focused on recognizing daily activities such as ‘walking’, ‘driving’, ‘biking’ etc. from smartphone sensors such as the accelerometer and GPS. However, a real world recommender system should be capable of recommending a diverse set of activities to users in addition to such diurnal activities. To address this, we employ an activity hierarchy which consists of lifestyle, recreational and tourist activities for activity recommendations.

Figure 2 shows a partial view of our Activity Hierarchy. We generated it by combining popular activities from the Yelp category list, Viator things to do list, and the FourSquare venue category hierarchy. Our hierarchy has a tree structure of depth 4 and contains 120 activity nodes including the root node ‘Personal Life’. There are 4 high-level (depth 1) activities which branch out into 15 coarse-grained activities (depth 2). All the high-level and some of the coarse-grained activities that they include are shown in Figure 2. Some of the coarse-grained activities at depth 2 are further categorized into 28 fine-grained activities (depth 3) which are further categorized into 128 leaf node activities. For instance, the root node ‘Personal Life’ branches out into coarse-grained activities including ‘Leisure’ which is further categorized into fine-grained activities such as ‘Entertainment’ and ‘Recreation’ etc. ‘Entertainment’ is further categorized into leaf node activities such as ‘Music’, ‘Movies’ and ‘Dance’. It is possible that a user is engaged in an activity which is not present in our hierarchy. To address this, we have a generic category ‘Other’ at each depth.

Inferring user’s activities from our data set is non trivial and challenging. Unlike previous research, we do not have labeled sensory data from users’ phones. Though all the photos in our dataset are geotagged, a user could be engaged in several probable activities at a location. Also, less than 50% of the photographs are annotated with user generated photo content such as a detailed name or a description and even fewer photographs have tags or comments which can provide some indication of the activity occurring when the photograph was taken. In addition, crowdsourcing the labeling of activities for 588,000 photos is not feasible.

To address these challenges, we propose an automated and unsupervised NLP based algorithm (Algorithm 2) to infer a user’s activity from user generated text such as the photo content items. As shown, we first remove all stop words from each photo content item and concatenate the items. We then perform a web search query to elucidate the meaning of the concatenated items and retrieve the

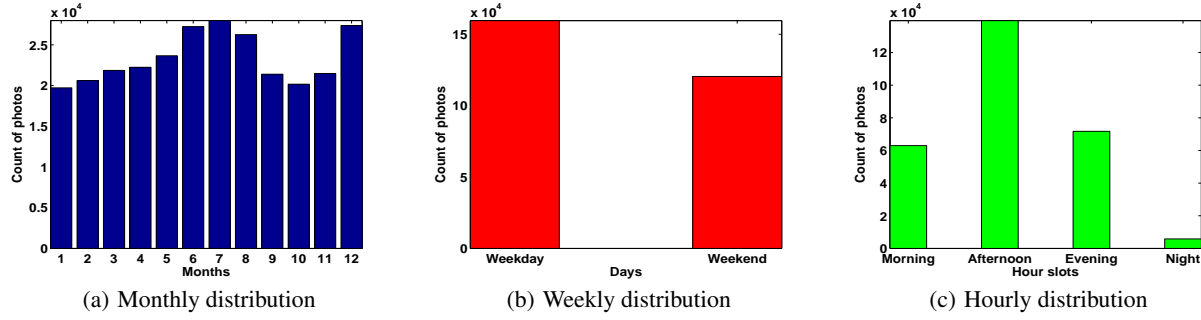


Figure 3: Monthly, Weekly and Hourly distribution of photographs in the San Francisco Bay Area dataset

Algorithm 2: Algorithm for activity inference from user generated text

Input: Photo content items such as name, description, tag and comments and Activity Hierarchy

Output: Inferred Activities

Remove stop words from each photo content item;
 Concatenate the content items to generate a search query;
 Perform a web search using the search query and retrieve the text content of the top web search result;
 Extract features such as named entities and types, document categories and social tags from the text content;

foreach Activity in the Activity Hierarchy **do**
 | Compute Semantic Relatedness (SR) scores between the features and the Activity;
 | MaxSRScore for each activity $\leftarrow \arg \max$ (SR Score between any of the features and the activity);
 | **if** MaxSRScore $< SR_{threshold}$ **then**
 | | MaxSRScore $\leftarrow 0.0$;
end

MAXMaxSRScore $\leftarrow \arg \max$ (MaxSRScore);
if MAXMaxSRScore $\neq 0$ **then**
 | MaxActivities \leftarrow Activities with MAXMaxSRScore;
if MaxActivities = \emptyset **then**
 | MaxActivities \leftarrow Propagated photo labels based on distance and time;
return MaxActivities;

content of the topmost web search result. From this content, we extract features such as named entities, document categories and tags. These features are extracted using three NLP techniques:

- Named Entity Recognition - a subtask of information extraction that identifies names of persons, organizations etc. in a given text or sentence
- Document Categorization - a task that classifies the subject or topic of the text, and
- Social Tagging - the practice of generating tags or keywords by users rather than experts to describe online content.

For this, we employ a tool called OpenCalais⁵ which can recognize up to 39 entities from the text. It also categorizes the text into one or more 18 document categories such as Finance, Entertainment etc. In addition, it associates one or more social or common sense tags with the text. The use of these techniques ensures that a large amount of world knowledge is exploited for feature extraction.

We then compute the Semantic Relatedness (SR) scores between each activity and each feature extracted from the web content. Se-

mantic Relatedness⁶ is a metric for determining the similarity of two documents or phrases based on their semantic meaning. It is normalized to a value between 0 (little to no relatedness) and 1 (extremely high relatedness). While there are several techniques and systems available for computing Semantic Relatedness, we employ the Semantic Textual Similarity (STS) system [15] for computing the SR scores. STS is based on Latent Semantic Analysis (LSA) along with WordNet knowledge and is trained on the LDC Gigawords and Stanford Webbase corpora. Harispe et al. [16] present a detailed discussion of semantic relatedness metrics and techniques.

For each activity, we store the maximum SR score (MaxSRScore) between any of the features and the activity. If the MaxSRScore for an activity is less than a threshold SR score, we set it to 0.0. This helps in reducing noise and false positives. Since Semantic Relatedness is a cosine similarity measure, a threshold of 0.293 ($1 - \cos 45^\circ$) is generally considered an appropriate threshold, and we use that in our current implementation. Finally, we iterate over all the activities and select those with the highest MaxSR scores. If a photo has no labeled activities, we apply a simple label propagation technique to label it based on its nearest neighbors (with respect to location and time). This is intuitive because if two consecutive photos are close in time and location, it is highly probable that the user was performing the same activity when they were taken.

To illustrate Algorithm 2 better, consider a photo which has been tagged as ‘Brazen Wildcat Half’ by the user. This phrase by itself does not convey any meaningful information about the user’s activity but a web search for it reveals content such as ‘Brazen Wildcat Half Marathon Racing...’. From the web content text, we get features such as ‘Trail Run’, ‘Athletics’ and ‘Recreation’ etc. Algorithm 2 then maps these features to the activities: Sports, Running and Recreation. From this, we can infer that the user was engaged in these activities when the photo was taken.

4.3 Time hashing

Since each photo has a unique timestamp, the number of timestamps in our dataset is huge. To address this, we perform feature hashing to hash the value of each timestamp to a timeslot. In order to determine the granularity of timeslots, we analyzed the monthly, weekly, daily and hourly distribution of photographs.

Figure 3(a) shows the distribution of photographs in the San Francisco Bay Area dataset for months in a year (1= January until 12 = December). As evident, the number of photos varies with each month. Clearly, July and December have the highest photo counts as these months have the typical vacations of July 4th and Christmas. We also analyzed the differences in weekdays and weekends. As shown in Figure 3(b), weekdays have a higher photo count.

We further considered the distribution of photographs at different hours in a day. To this end, we divided a day into 4 hourly slots:

⁶Semantic Relatedness and Semantic Similarity have often been used interchangeably in literature. We use Semantic Relatedness in this paper to refer to the metric.

⁵<http://www.opencalais.com/>

Grid Size r (m)	# of unique hashed locations		
	San Francisco	Las Vegas	Chicago
300	7869	2747	5082
500	5565	1932	3665
700	4222	1475	2855
1000	2999	1058	2070

Table 3: Number of unique hashed locations in our 3 datasets (number varies with grid size)

- Morning - hours between 6 am and 12 pm
- Afternoon - hours between 12 pm and 6 pm
- Evening - hours between 6 pm and 12 am
- Night - hours between 12 am and 6 am

As seen in Figure 3(c), the highest count of photographs is taken in the afternoon time slot i.e. between 12 pm and 6 pm. Intuitively, the least number (5834) were taken at night.

Based on this analysis, we generated buckets of hashed time slots. Since there are 12 months in a year, 2 types of days in a week - weekday and weekend, and each day has 4 hourly slots, the total number of hashed timeslots is 96. For each photo, we first convert its epoch timestamp to a standard date time format which is then mapped to a hashed timeslot. For instance, a photo taken on August 19, 2013 at 2 pm will be hashed to ‘WeekDay_8_Afternoon’.

5. OUR APPROACH

5.1 Data Modeling of various dimensions

After inferring the user, location, activity and time information from the Flickr photos, the data in each dataset needs to be modeled along the various dimensions of our multi-dimensional tensor. These dimensions are:

- Location dimension - We first apply data filtering to reduce noise. We retain only those unique hashed locations that have been visited by at least u_t users (in our experiments, we set $u_t = 5$). Thus, for each unique hashed location present in our 3 datasets, we determine the number of unique users from that dataset who visited that location. We remove all locations that have ≤ 5 unique users. We perform this thresholding mainly to eliminate locations that can be residences of users or random spots such as roads. Table 3 shows the final count of locations in our datasets. The number of locations vary with the hashed grid size. These locations correspond to the Location dimension of our 4 dimensional tensor.
- User dimension - Once the final set of locations for each dataset has been obtained, we use only those photographs which have been taken at these locations. The corresponding user IDs for these photographs represent the User dimension. Our final set of users consists of 2498 unique users in the San Francisco Bay Area, 573 unique users in the Las Vegas and 1850 unique users in the Chicago datasets.
- Activity dimension - The 120 activities in our activity hierarchy represent the Activity dimension of the tensor.
- Time dimension - The 96 hashed time slots represent the Time dimension of the tensor.

5.2 Constructing the tensors and matrices

5.2.1 User \times Location \times Activity \times Time tensor

The four dimensional (User, Location, Activity and Time) data modeled from each of the Flickr datasets can be represented as a sparse tensor $X \in \mathbb{R}^{u \times l \times a \times t}$ where u is the # of users, l is the # of locations, a is the # of activities and t is the # of time slots. The ratings placed into this tensor should represent the user’s interest for performing a certain activity at a certain location at a certain time. However, in our data set users do not provide any explicit ratings.

We thus derive an implicit feedback [32] value normalized over [0.0, 1.0]. Each cell value of the tensor represents the frequency of the current user being at the current location performing the current activity at the current time, which serves as implicit feedback. The counts are further normalized based on the total number of data points (photographs) for each user.

Moreover, each user typically visits a small subset of the possible locations at only a few of all the possible times and performs a fraction of all the activities possible. Hence, our tensor is very sparse and any given fiber will have only a few non-empty entries. After this construction, the tensors from the 3 datasets have a density of the order of $10^{-3}\%$. Thus, the tensors are 99.999% sparse.

5.2.2 Location \times Activity Matrix

Knowing what activities occur in a given location can enable inference of the activity a user is engaged in when he is at the location. As mentioned earlier, even though all the photographs have a geolocation, more than 50% of the photographs do not have any photo content to indicate what activity the user could be engaged in. Hence, this location-activity information can enable inference of the most likely activity a user could be engaged in at a location.

To obtain this, for each activity in our hierarchy, we query the Foursquare location database to find all the locations in each of our datasets, where that particular activity can occur. Thus, for each location l_i in a dataset, we get an n -dimensional frequency vector $c_i = [c_1, c_2 \dots c_n]$ for n activities (where $n = 120$). Each $c_{i,j}$ is normalized as $\frac{c_{i,j}}{\sum_{j=1}^n c_{i,j}}$. From this information, we construct a

Location \times Activity matrix $Y \in \mathbb{R}^{l \times a}$ which contains normalized counts for the activities that occur in each location.

5.2.3 Location \times Venue Matrix

Activities typically occur at a venue or a POI such as a restaurant, shopping mall, etc. For instance, a user would ‘Eat’ at a ‘Restaurant’ or ‘Shop at a ‘Shopping Mall’ in a location. Hence, the knowledge of venues that are present in a location can be leveraged to enable the inference of the user’s activity.

For each location in each of our datasets, we also obtain the counts of different venues (from the Foursquare Location database) that are present in the location. The venues belong to the Foursquare Venue Hierarchy that includes 470 different types of venues such as restaurants, movie theaters, museums etc. Thus, for each location l_i in a dataset, we get an m -dimensional frequency vector $v_i = [v_1, v_2 \dots v_m]$ for m venues (where $m = 470$). Each $v_{i,j}$ is normalized as $\frac{v_{i,j}}{\sum_{j=1}^m v_{i,j}}$. From this information, we construct a Location \times

Venue matrix $J \in \mathbb{R}^{l \times v}$ which contains normalized counts for the venues present in each location.

5.2.4 Location \times Location Similarity Matrix

The locations that have similar type and count of venues will host similar activities. Hence, we employ the Location \times Venue matrix to compute the location similarity information. For each pair of locations l_i and l_j in each dataset, we calculate the cosine similarity between the venues vectors as $\text{sim}(l_i, l_j) = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|}$ where $0 \leq \text{sim}(l_i, l_j) \leq 1$. Using the location similarity information, we construct a symmetric Location \times Location matrix $Z \in \mathbb{R}^{l \times l}$

5.2.5 Activity \times Activity Correlation Matrix

Users may often have preferences for activities that are similar and correlated. For instance, a user who likes sports might engage in several different types of outdoor sports such as basketball, tennis etc. Hence, this knowledge of correlation between activities can be exploited easily to further boost the information about the kinds of activities a user could be interested in.

We use the SR metric to compute correlation between all the activities present in our hierarchy. For each pair of activities a_i and a_j in our Activity hierarchy, we calculate the SR score between their descriptions to get $\text{sim}(a_i, a_j)$. For instance, SR score between 'Basketball' and 'Tennis' is 0.85, indicating that they are highly correlated. This is intuitive as both are outdoor recreational sports. From the Activity - Activity correlation information, we construct a symmetric Activity \times Activity matrix $S \in \mathbb{R}^{a \times a}$.

As evident, each of the 2 dimensional matrices has at least one mode in common with the 4 dimensional tensor.

5.3 Objective function formulation

We formulate the objective function for our multi-dimensional recommendation problem to simultaneously factorize these coupled tensors and matrices. The objective function is:

$$F = \frac{1}{2} \|W \times (X - U \circ L \circ A \circ T)\|^2 + \frac{\lambda_1}{2} \|Y - LA^\top\|^2 + \frac{\lambda_2}{2} \|S - AA^\top\|^2 + \frac{\lambda_3}{2} \|Z - LL^\top\|^2 + \frac{\lambda_4}{2} \|TR\|^2 + \frac{\lambda_5}{2} (\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$$

Thus, our objective function comprises six summands and can be written in the form $F = F_1 + F_2 + F_3 + F_4 + F_5 + F_6$. The six summands and the terms and symbols that they include are:

- F_1 - The weighted least squares error term for the decomposition of the 4 dimensional tensor X into the factor matrices $U \in \mathbb{R}^{u \times k}$, $L \in \mathbb{R}^{l \times k}$, $A \in \mathbb{R}^{a \times k}$ and $T \in \mathbb{R}^{t \times k}$ where k is the number of factors. W is a weight tensor $\in \mathbb{R}^{u \times l \times a \times t}$ and indicates the missing entries in X such that:

$$w_{u,l,a,t} = \begin{cases} 1 & \text{if } x_{u,l,a,t} \text{ is known} \\ 0 & \text{if } x_{u,l,a,t} \text{ is missing} \end{cases}$$

This term tries to minimize the loss in only the known entries of the tensor.

- F_2 - The least squares error term for the decomposition of the 2 dimensional matrix Y into the factor matrices $L \in \mathbb{R}^{l \times k}$ and $A \in \mathbb{R}^{a \times k}$
- F_3 - The least squares error term for the decomposition of the 2 dimensional symmetric matrix S into the factor matrices $A \in \mathbb{R}^{a \times k}$ and its transpose
- F_4 - The least squares error term for the decomposition of the 2 dimensional symmetric matrix Z into the factor matrices $L \in \mathbb{R}^{l \times k}$ and its transpose
- F_5 - Regularization term for temporal smoothing. It leverages the fact that human behavior in successive time periods will be similar and will have a gradual variation. Hence, it tries to reduce the error between consecutive time slots. R is a bi-diagonal matrix $\in \mathbb{R}^{k \times k}$ with 1 on the main diagonal and -1 on the diagonal above it.
- F_6 - Regularization term to avoid overfitting.
- $\lambda_1 - \lambda_5$ are model parameters.
- $\|\cdot\|^2$ denotes the Frobenius norm, \circ denotes the outer product

In general, there is no closed form solution for F , so we use numerical methods, such as gradient descent, to solve this problem. By using the representations in [21], we take the first order derivatives of F with respect to each of the factors to get the following:

$$\begin{aligned} \nabla_U F &= (W^{(1)} - X^{(1)})(T * A * L) + \lambda_5 U \\ \nabla_L F &= (W^{(2)} - X^{(2)})(T * A * U) + \lambda_1 (LA^\top - Y)A + \lambda_3 (-Z - Z^\top + 2LL^\top)L + \lambda_5 L \\ \nabla_A F &= (W^{(3)} - X^{(3)})(T * L * U) + \lambda_1 (LA^\top - Y)^\top L + \lambda_2 (-S - S^\top + 2AA^\top)A + \lambda_5 A \\ \nabla_T F &= (W^{(4)} - X^{(4)})(A * L * U) + \lambda_4 TR + \lambda_5 T \end{aligned}$$

where $W^{(i)}$ and $X^{(i)}$ denotes the mode- i tensor unfolding or matricization⁷ of W and X such that $W^{(1)} \in \mathbb{R}^{u \times lat}$, $W^{(2)} \in \mathbb{R}^{l \times uat}$, $W^{(3)} \in \mathbb{R}^{a \times ult}$, and $W^{(4)}$ and $X^{(4)} \in \mathbb{R}^{t \times ula}$, and $*$ denotes the Khatri-Rao product.

⁷Matricization involves generation of the matrix representation of a tensor in which all the column or row matrices are stacked one after the other.

Algorithm 3: Algorithm for minimizing objective function using gradient descent

Input: Sparse tensor $X \in \mathbb{R}^{u \times l \times a \times t}$, 2 D matrices $Y \in \mathbb{R}^{l \times a}$, $Z \in \mathbb{R}^{l \times l}$, $S \in \mathbb{R}^{a \times a}$, k and convergence criteria

Output: Complete tensor $M \in \mathbb{R}^{u \times l \times a \times t}$

for $n = 1 : \text{size}(X)$ **do**

 Initialize U, L, A, T ;

end

Initialize $F, \nabla_U F, \nabla_L F, \nabla_A F$ and $\nabla_T F$;

Set $i = 0$;

while *not converged* **do**

 Compute step length α_i ;

 Compute the gradients $\nabla_U F_i, \nabla_L F_i, \nabla_A F_i, \nabla_T F_i$;

$U_{i+1} = U_i - \alpha_i \nabla_U F_i, L_{i+1} = L_i - \alpha_i \nabla_L F_i, A_{i+1} = A_i -$

$\alpha_i \nabla_A F_i, T_{i+1} = T_i - \alpha_i \nabla_T F_i$;

 Compute F_{i+1} ;

end

$M \leftarrow U \circ L \circ A \circ T$;

return M ;

5.4 Minimizing the objective function

We employ Algorithm 3, which uses gradient descent, to minimize F and its gradient G . We implemented it in MATLAB using the Tensor Toolbox [4]. As input, the algorithm takes the incomplete sparse tensor (X), the low dimensional matrices (Y, Z , and S), the number of factors (k) as well as the stopping or convergence criteria. The individual components (U, L, A, T) are initialized using the n -mode singular vectors of the tensor X which span the subspace of the mode- n fibers i.e. the left singular vectors of the n -mode matricization of X . In each iteration, we first compute the step length using the More-Thuente line-search method [30]. This method tries to find a point that satisfies both the decrease and curvature conditions of the function by using bracketing as well as quadratic and cubic interpolation. We then compute the values of the gradients for the objective function and the components, and update the objective function value by taking a step in the direction of the gradient.

The convergence criteria are set as:

- Relative change in Function Value i.e. $\frac{F_{i+1} - F_i}{F_i} \leq 10^{-10}$
- Relative change in Gradient Value i.e. $\frac{G_{i+1} - G_i}{G_i} \leq 10^{-10}$
- Number of iterations $i \leq 10^5$

Finally, when the algorithm converges, we obtain the complete tensor $M \in \mathbb{R}^{u \times l \times a \times t}$ by taking the outer product of the individual components U, L, A and T .

5.5 Discussion

We note that while we have focused on $N=4$ dimensions in this paper, we can easily extend the model to accommodate additional dimensions, without loss of generality, by performing data engineering and modifying the objective function. For instance, if we add another contextual dimension such as users' purchases, we can supplement the sparse tensors and matrices with User \times Purchases or Location \times Purchases matrix obtained from another data source. We can then modify the objective function to include the purchases dimension and employ Algorithm 3 to minimize it.

6. EVALUATION

6.1 Methodology

To evaluate the recommendations produced by our system, we used the following methodology: we first randomly split each of

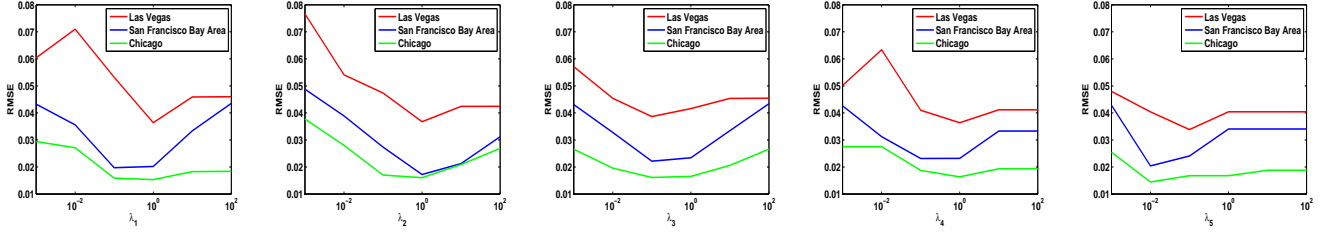


Figure 4: Variation of RMSE for different values of $\lambda_1 - \lambda_5$ on the validation sets

the three datasets (from San Francisco Bay Area, Las Vegas and Chicago) into training and testing sets with a 70:30 ratio, where we use the training set for training and tuning the model parameters. We use the held-out test set for computing the performance metrics over the predicted and ground truth values. More formally, we define P to be a test dataset containing n values. For each held out value $\in P$, y_i denotes the ground truth value and \hat{y}_i denotes the predicted value.

6.2 Metrics

We use three standard performance metrics [18] for evaluating the performance of our approach on a test set P :

- Root Mean Squared Error (RMSE) - RMSE is computed as

$\sqrt{\frac{\sum_{i=1}^n \hat{y}_i - y_i}{n}}$. However, RMSE can be susceptible to large errors and often places more emphasis on them. Hence, we compute Mean Absolute Error (MAE) as well to evaluate the performance of our approach.

- Mean Absolute Error (MAE) - MAE is computed as $\frac{\sum_{i=1}^n |\hat{y}_i - y_i|}{n}$. However, both RMSE and MAE may be less appropriate for tasks where a ranked result is returned to the user, who then only views items at the top of the ranking. For this, we compute Normalized Discounted Cumulative Gain (nDCG).
- Normalized Discounted Cumulative Gain (nDCG) - nDCG is commonly used in information retrieval to measure a search engine's performance. A higher nDCG value for a list of search results means that, the highly relevant items appeared at the top (with higher ranks) in the result list. In particular, nDCG[p], often referred to as nDCG@p, measures the relevance of top p results and is defined as:

$$\text{nDCG@p} = \frac{\text{DCG@p}}{\text{IDCG@p}}$$
where $\text{DCG@p} = \text{rel}_1 + \sum_{i=2}^p \frac{\text{rel}_i}{\log_2 \text{rel}_i}$,
IDCG@p is the DCG@p value of ideal ranking list and rel_i is a relevance value. nDCG ranges from 0 to 1. The higher the nDCG value is, the better a ranking result list is.

6.3 Parameter Tuning

We performed parameter tuning using parameter-sweeping experiments for the different training sets. We randomly split each training set into training and validation sets with a 4:1 ratio. We held out the validation set and constructed the model using the training set with different values for the model parameters, $\lambda_1 - \lambda_5$, # of factors (k) and the hashed location grid sizes. We then computed RMSE on the held out validation sets and picked the parameter values that minimized it.

6.3.1 Impact of model parameters

For tuning each individual parameter, we set the remaining parameters as 0.00001 in order to reduce their impact on the model performance. We then ran the parameter-sweeping experiments 5

Grid Size r (m)	RMSE		
	San Francisco	Las Vegas	Chicago
300	0.0217	0.0378	0.0170
500	0.0213	0.0376	0.0162
700	0.0222	0.0392	0.0175
1000	0.0224	0.0437	0.0186

Table 4: RMSE on validation set for various location grid sizes

times for each parameter value and averaged the RMSE. Figure 4 shows the variation of RMSE for different values of the model parameters for the 3 validation sets. As evident, the RMSE first increases and later decreases as value of each parameter ($\lambda_1 - \lambda_5$) increases. This is because when a parameter value is too small, the model cannot fully utilize the information from the corresponding matrix. On the other hand, if it is too large, then the information from the matrix will dominate the objective function. After this tuning, we set $\lambda_1 = 1.0$, $\lambda_2 = 1.5$, $\lambda_3 = 0.5$, $\lambda_4 = 1.0$ and $\lambda_5 = 0.02$.

6.3.2 Impact of number of factors

We varied the number of latent factors (k) from 10 - 50. However, we observed that the RMSE did not exhibit significant variation which implies that changing the number of factors did not have a significant impact on performance, as also reported by Zheng et al. [40, 41]. For our experiments, we set $k = 30$.

6.3.3 Impact of location grid size

Table 4 shows the RMSE for the 3 validation datasets for different location grid sizes. The grid size of 500m has the lowest RMSE for all the validation sets and hence we use that for our experiments. This is also intuitive because recommended locations within a distance of 500m can be easily reached on foot.

Henceforth, all experiments will use the above mentioned parameters.

6.4 Comparison with Baselines

We further compare our approach with 7 state of the art baselines.

6.4.1 Collaborative Filtering baselines

These baselines exploit similarity on each of the individual dimensions to complete the tensor. They take only the 4 dimensional tensor as input. We implemented 4 CF baselines:

- User-User Collaborative Filtering baseline (UCF) - This baseline exploits the user-user similarity information to fill in the missing entries of the tensor. In particular, for UCF, we consider CF on each user \times location matrix with respect to each activity and each time slot, on each user \times activity matrix with respect to each location and each time slot, and on each user \times time matrix with respect to each location and each activity independently. To this end, we matricize the sparse tensor X in Mode 1 to generate matrix $X^{(1)} \in \mathbb{R}^{u \times lat}$. We

Approach	RMSE		
	San Francisco	Las Vegas	Chicago
Our approach	0.0197	0.0339	0.0153
UCF	0.0324	0.0486	0.0293
LCF	0.0336	0.0430	0.0311
ACF	0.0333	0.0433	0.0317
TCF	0.0319	0.0484	0.0315
Standard CP	0.0224	0.0427	0.0178
HOSVD	0.0227	0.0405	0.0175
ALS	0.0222	0.0389	0.0173

Table 5: RMSE for our approach and baselines on the 3 test sets

Approach	MAE		
	San Francisco	Las Vegas	Chicago
Our approach	0.0076	0.0179	0.0049
UCF	0.0104	0.0314	0.0062
LCF	0.0102	0.0310	0.0066
ACF	0.0105	0.0323	0.0069
TCF	0.01	0.0354	0.0065
Standard CP	0.0095	0.0228	0.0058
HOSVD	0.0098	0.0239	0.0061
ALS	0.0093	0.0218	0.0054

Table 6: MAE for our approach and baselines on the 3 test sets

then use Pearson correlation coefficient between the vectors in the matrix to compute pairwise user similarity information. For each user, we compute the weighted average of the top N similar users to predict the missing values.

- Location-Location Collaborative Filtering baseline (LCF) - Similarly, the LCF baseline exploits the location-location similarity information to fill in the missing entries of the tensor. In this case, we matricize the sparse tensor X in Mode 2 to generate matrix $X^{(2)} \in \mathbb{R}^{l \times uat}$. We then use Pearson correlation coefficient between the vectors in the matrix to compute pairwise location similarity information. For each location, we then compute the weighted average of the top N similar locations to predict the missing values.
- Activity-Activity Collaborative Filtering baseline (ACF) - The ACF baseline exploits the activity-activity similarity information to fill in the missing entries of the tensor. In this case, we matricize the sparse tensor X in Mode 3 to generate matrix $X^{(3)} \in \mathbb{R}^{a \times ult}$. We then use Pearson correlation coefficient between the vectors in the matrix to compute pairwise activity similarity information. For each activity, we then compute the weighted average of the top N similar activities to predict the missing values.
- Time-Time Collaborative Filtering baseline (TCF) - Finally, the TCF baseline exploits the time-time similarity information to fill in the missing entries of the tensor. In this case, we matricize the sparse tensor X in Mode 4 to generate matrix $X^{(4)} \in \mathbb{R}^{t \times ula}$. We then use Pearson correlation coefficient between the vectors in the matrix to compute pairwise time similarity information. For each time slot, we then compute the weighted average of the top N similar time slots to predict the missing values.

In these experiments, we set $N = 10$, since the prediction results do not depend on N significantly as suggested by Zheng et al. [40]

6.4.2 Model based baselines

We implemented the standard CP decomposition model which, when applied to our multi-dimensional recommendation problem, has an objective function of the form:

$$F = \frac{1}{2} \|X - U \circ L \circ A \circ T\|^2 + \frac{\lambda_5}{2} (\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$$

Thus, it takes only the 4 dimensional tensor as input and its objective function has only the tensor decomposition term along with the regularization term. This can be obtained by setting $\lambda_1 - \lambda_4 = 0$ in our objective function (see Section 5.3) and replacing the weighted least squares error term with the standard least squares error term in summand F_1 . We then minimize this objective function using the gradient descent based Algorithm 3.

6.4.3 Algorithmic baselines

The 2 algorithmic baselines that we employ are:

- Higher Order Singular Value Decomposition (HOSVD) - We implement the HOSVD approach proposed by Lathauwer et al. [12]. This approach also takes only the 4 dimensional tensor as input. It first matricizes the 4 dimensional User \times Location \times Activity \times Time tensor X along each of the 4 modes to get the matrices $X^{(1)}, X^{(2)}, X^{(3)}$ and $X^{(4)}$. On each matrix, SVD is applied to compute the low rank approximation: $X^{(i)} = U^{(i)} \cdot S^{(i)} \cdot V^{(i)\top}$ where $1 \leq i \leq 4$. The core tensor S is then constructed as:
 $S = X \times_1 U_{c_1}^{(1)\top} \times_2 U_{c_2}^{(2)\top} \times_3 U_{c_3}^{(3)\top} \times_4 U_{c_4}^{(4)\top}$
where $U_{c_1}^{(1)\top}, U_{c_2}^{(2)\top}, U_{c_3}^{(3)\top}$, and $U_{c_4}^{(4)\top}$ are the transposes of the c_1 -dimensionally reduced $U^{(1)}$ matrix, c_2 -dimensionally reduced $U^{(2)}$ matrix, c_3 -dimensionally reduced $U^{(3)}$ matrix, and c_4 -dimensionally reduced $U^{(4)}$ matrices respectively. Finally, the completed matrix M is obtained as:
 $M = S \times_1 U_{c_1}^{(1)} \times_2 U_{c_2}^{(2)} \times_3 U_{c_3}^{(3)} \times_4 U_{c_4}^{(4)}$
where c_1, c_2, c_3 and c_4 are set empirically. Based on the experiments of Nanopoulos [31], we preserve 30% of the information in each matrix.
- Alternating Least Squares (ALS) - In ALS, the objective function is a standard CP formulation and the idea is to solve for each factor matrix, leaving all other factors fixed. We implement the ALS algorithm proposed by Kolda and Bader [21, 22] for the standard CP decomposition of our multi-dimensional recommendation problem.

6.5 Results

For testing experiments, we held out the test dataset and generated the completed tensor using the training dataset with the tuned parameter values, # of factors k (30) and for the optimal grid size (500m). We then computed the RMSE and MAE between the predicted and the ground truth held out values of the test set.

Since we do not have human supplied relevance rankings, we compute nDCG on the held out known values. We calculated nDCG with respect to each recommendation dimension by fixing the remaining dimensions and computing nDCG on the current dimension. Thus, for each of the held out values in the test set, we first fixed user, location and activity and ranked the time slots in the completed tensor. This ranking was used to calculate DCG (refer to Section 6.2) for time. To compute iDCG for time, we determined the ranking for the ground truth time values. We then calculated nDCG@p (with $p = 5$) for time. Finally, we averaged the values for all user, location and activity combinations to generate an averaged nDCG@5 for the time dimension. Similarly, we fixed user, location and time values and ranked activities to compute nDCG@5 for the activity dimension and fixed user, activity and time to compute nDCG@5 for the location dimension.

Tables 5, 6, and 7 show the results achieved by our approach as well as the baselines on the 3 test datasets for the 3 performance metrics: RMSE, MAE and nDCG@5. For RMSE and MAE, a lower value signifies superior performance while for nDCG, a higher value signifies superior performance. For each metric, we ran all algorithms 5 times on each test dataset and averaged the result. Clearly, our approach outperforms these baselines.

Approach	nDCG@5 for location		
	San Francisco	Las Vegas	Chicago
Our approach	0.898	0.835	0.821
UCF	0.691	0.523	0.693
LCF	0.71	0.514	0.682
ACF	0.702	0.582	0.695
TCF	0.735	0.593	0.701
Standard CP	0.685	0.453	0.651
HOSVD	0.890	0.815	0.798
ALS	0.798	0.722	0.748
Approach	nDCG@5 for activity		
	San Francisco	Las Vegas	Chicago
Our approach	0.869	0.827	0.798
UCF	0.452	0.465	0.467
LCF	0.457	0.477	0.485
ACF	0.440	0.458	0.412
TCF	0.391	0.34	0.401
Standard CP	0.341	0.314	0.393
HOSVD	0.829	0.817	0.714
ALS	0.770	0.737	0.692
Approach	nDCG@5 for time		
	San Francisco	Las Vegas	Chicago
Our approach	0.833	0.832	0.807
UCF	0.581	0.612	0.605
LCF	0.563	0.605	0.567
ACF	0.541	0.609	0.575
TCF	0.593	0.632	0.582
Standard CP	0.578	0.605	0.563
HOSVD	0.812	0.82	0.798
ALS	0.732	0.725	0.656

Table 7: nDCG (for location, activity and time) for our approach and baselines on the 3 test sets

6.6 Discussion of Results

As evident from the results, the neighborhood based Collaborative Filtering baselines exhibit poor performance with respect to all the 3 performance metrics. There are two possible reasons for this - first, these approaches employ only the User \times Location \times Activity \times Time tensor X . Since the tensor is extremely sparse, computing similarity along any dimension will be error prone as most of the entries are missing. Hence, our approach which supplements the sparse tensor with additional information from external sources overcomes this hurdle. Second, each of these baselines predicts the missing values based on similarity along one dimension only, ignoring the other dimensions. Since our problem involves collaborative recommendations along multiple dimensions, it is important to employ all the dimensions for recommendations as we do.

The model based and algorithmic baselines such as Standard CP, HOSVD and ALS also utilize only the sparse tensor as input. As pointed out in Section 2, the standard CP approach demonstrates an increase in error if the data is extremely sparse. Similarly, ALS has poor convergence if the data is sparse and does not scale to large datasets. Moreover, the algorithmic baselines HOSVD and ALS have high space complexity. On the other hand, our approach overcomes the sparsity of the data by supplementing the sparse tensor with coupled matrices and also scales to large datasets. Hence, it outperforms these baselines.

Also, we note that the nDCG values for location varies greatly for each dataset for each algorithm while the nDCG values for activity and time do not exhibit such a high variation. This is possibly because the number of locations in each of the dataset varies significantly while the number of activities and time slots are constant for

Approach	Time (s)		
	San Francisco	Las Vegas	Chicago
Without weighting	12000	2465	10500
With weighting	9600	1180	8580

Table 8: Runtime comparison

each dataset. In addition, the nDCG values for location are higher in general for most of the algorithms and datasets, as compared to activity and time. This indicates that location is the most important dimension for recommendation, followed by time and then activity. This is also intuitive because in our datasets, each photo has a location and timestamp but $< 50\%$ photos have meta data to enable activity inference. Hence, the activity data is sparser than for location and time. This would affect the performance of all the baselines since they use the sparse tensor as the only input. However, in our approach, we supplement this sparse tensor with additional matrices involving Location, Activity and Time. We also perform regularized temporal smoothing. Hence, the nDCG for Activity and Time are higher and comparable to that of Location for our approach.

6.7 Runtime Comparison

Since our data is very sparse, we performed a weighted decomposition of the tensor which minimizes the error in only the known entries of the tensor. We also experimented without the weighting imposed on the tensor which minimizes the error in all the entries of the tensor. In this case, our objective function is:

$$F = \frac{1}{2} \|X - U \circ L \circ A \circ T\|^2 + \frac{\lambda_1}{2} \|Y - LA^T\|^2 + \frac{\lambda_2}{2} \|S - AA^T\|^2 + \frac{\lambda_3}{2} \|Z - LL^T\|^2 + \frac{\lambda_4}{2} \|TR\|^2 + \frac{\lambda_5}{2} (\|U\|^2 + \|L\|^2 + \|A\|^2 + \|T\|^2)$$

We minimized this function using Algorithm 3. Both the approaches have the same performance with respect to RMSE, MAE and nDCG@5 for various dimensions. However, they differ in their runtime. Table 8 compares their runtime performance. All experiments were run on a 64 bit Windows machine with core i7 processor and 16 GB RAM. As shown, the weighted approach is faster as it minimizes the loss on only the known entries of the tensor.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented a system and an approach for performing multi-dimensional collaborative recommendations for Who (User), What (Activity), When (Time) and Where (Location), using tensor factorization on sparse user generated data. We modeled our multi-dimensional recommendation problem as a joint analysis of a sparsely populated User \times Location \times Activity \times Time tensor along with several matrices which share one or more common modes with the tensor. These tensors and matrices have been constructed by fusing data for the various dimensions (users, locations, activities and time) from multiple heterogeneous data sources such as Flickr (a photo sharing website), Foursquare (a location based social network) Yelp (a crowd-sourced reviews based website) and Viator (a travel website). We factorized these tensors and matrices simultaneously using gradient descent. We evaluated our system and approach on large-scale real world data sets consisting of 588,000 Flickr photos collected from three major cities in the USA - San Francisco Bay area, Las Vegas and Chicago. We compared our approach with several state of the art baselines and demonstrated that it outperforms all of them. Our approach also demonstrated an improvement in runtime without the need for sacrificing performance.

We now plan to create a Hadoop-based pipeline for performing multi-dimensional analytics. We will also work on scaling this approach to larger and more complex real-time data sets. In addition, we plan to parallelize our algorithms and approach to run on higher-dimensional data.

8. REFERENCES

- [1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *SDM10: Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 701–712, 2010.
- [2] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.
- [3] E. Acar, T. G. Kolda, and D. M. Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. In *MLG’11: Proceedings of Mining and Learning with Graphs*, August 2011.
- [4] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. Available online, January 2012.
- [5] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 199–208. ACM, 2012.
- [6] V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M. W. Newman, K. Partridge, B. Price, et al. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1157–1166. ACM, 2008.
- [7] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.
- [8] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 316–322. IEEE, 2005.
- [9] X. Cao, G. Cong, and C. Jensen. Mining significant semantic locations from gps data. In *Proceedings of VLDB*, pages 1009–1020, 2010.
- [10] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.
- [11] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world’s photos. In *Proceedings of the 18th International Conference on World Wide Web*, pages 761–770. ACM, 2009.
- [12] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [13] N. Ducheneaut, K. Partridge, Q. Huang, B. Price, M. Roberts, E. H. Chi, V. Bellotti, and B. Begole. Collaborative filtering is not enough? experiments with a mixed-model recommender for leisure activities. In *User Modeling, Adaptation, and Personalization*, pages 295–306. Springer, 2009.
- [14] F. Girardin, J. Blat, F. Calabrese, F. Dal Fiore, and C. Ratti. Digital footprinting: Uncovering tourists with user-generated content. *Pervasive Computing*, 7(4):36–43, Oct.-Nov. 2008.
- [15] L. Han, A. Kashyap, T. Finin, J. Mayfield, and J. Weese. Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, 2012.
- [16] S. Harispe, S. Ranwez, S. Janaqi, and J. Montmain. Semantic measures for the comparison of units of language, concepts or instances from text and knowledge representation analysis. *arXiv preprint arXiv:1310.1285*, 2013.
- [17] R. A. Harshman. Foundations of the parafac procedure: models and conditions for an “explanatory” multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(1), 1970.
- [18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
- [19] B. Hidasi and D. Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.
- [20] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.
- [21] T. G. Kolda. *Multilinear operators for higher-order decompositions*. United States. Department of Energy, 2006.
- [22] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.
- [23] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [24] L. Liao. *Location-based activity recognition*. PhD thesis, University of Washington, 2006.
- [25] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.
- [26] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of KDD*, pages 1043–1051. ACM, 2013.
- [27] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, 2010.
- [28] P. Melville and V. Sindhwani. Recommender systems. In *Encyclopedia of machine learning*, pages 829–838. Springer, 2010.
- [29] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008.
- [30] J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.
- [31] A. Nanopoulos. Item recommendation in collaborative tagging systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4):760–771, 2011.
- [32] D. W. Oard, J. Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83. Wollongong, 1998.
- [33] D. Quercia, R. Schifanellai, and L. M. Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of Hypertext*, pages 116–125. ACM, 2014.
- [34] M. Sattari, M. Manguoglu, I. H. Toroslu, P. Symeonidis, P. Senkul, and Y. Manolopoulos. Geo-activity recommendations by using improved feature combination. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 996–1003. ACM, 2012.
- [35] Y. Shi, P. Serdyukov, A. Hanjalic, and M. Larson. Nontrivial landmark recommendation using geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 4(3):17–37, June 2013.
- [36] A. P. Singh and G. J. Gordon. Relational learning via

- collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.
- [37] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.
- [38] J. D. Ullman, J. Leskovec, and A. Rajaraman. *Mining of Massive Datasets*. Cambridge University Press, 2011.
- [39] Y. Yang, Z. Gong, and L. Hou U. Identifying points of interest by self-tuning clustering. In *Proceedings of SIGIR*, pages 883–892. ACM, 2011.
- [40] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, pages 236–241, 2010.
- [41] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038. ACM, 2010.