Improved Trip Planning by Learning from Travelers' Choices

Boris Chidlovskii BCHIDLOVSKII@XRCE.XEROX.COM
Xerox Research Center Europe, 6 chemin Maupertuis,38240 Meylan, France

Abstract

We analyze the work of urban trip planners and the relevance of trips they recommend upon user queries. We propose to improve the planner recommendations by learning from choices made by travelers who use the transportation network on the daily basis. We analyze individual travelers' trips and convert them into pair-wise preferences for traveling from a given origin to a destination at a given time point. To address the sparse and noisy character of raw trip data, we model passenger preferences with a number of smoothed time-dependent latent variables, which are used to learn a ranking function for trips. This function can be used to re-rank the top planner's recommendations. Results of tests for cities of Nancy, France and Adelaide, Australia show a considerable increase of the recommendation relevance.

1. Introduction

Most cities and agglomerations around the world propose their trip planners, in the form of a web or mobile application. Upon a user travel request, they recommend trips using a static library of roads and public transportation network and services. Although these planners are increasingly reliable in their knowledge of transportation network and available services, they all share the same static-world assumptions. In particular, they make a general assumption of *constancy and universality* (Letchner et al., 2006), that the optimal trip is independent of the time of day of the actual journey and of the passengers' preferences.

In reality, constancy and universality rarely hold. Most urban travelers can verify that the best trip between work and home at midnight is not necessarily the best choice to make between the same locations at 8am. Similarly, different passengers may choose different ways to travel between the same origin and destination points.

While the personal knowledge plays an important role, in many cases passengers simply have different preferences about the trip planning. For example, one passenger may avoid multiple changes, by extending the duration of her journey by a few minutes, while another passenger simply wants to arrive as quickly as possible to the destination.

When a user queries a planner for a journey from origin o to destination d starting at time t_s, there are often a large number of trips satisfying the query. Planners are designed to provide the k-top recommendations according to a set of predefined criteria, such as the minimal transfer time, the minimal number of changes, etc.. Their work is similar to any information retrieval system, where the goal is to place the most relevant documents among the k-top answers. Therefore, it is highly desirable that a trip planner behaves intelligently and suggests k-top trips which reflect the real passengers' preferences.

In this paper we closely analyze the cases of divergence between the planner recommendations and real choices made by urban travelers. We collect two sets of individual trips extracted from fare collection systems in cities of Nancy, France and Adelaide, Australia (see Figure 1). We compare these data to the city planners' recommendations; and in the of case of divergence, we propose a novel method to rank the trips that better reflects the reality.

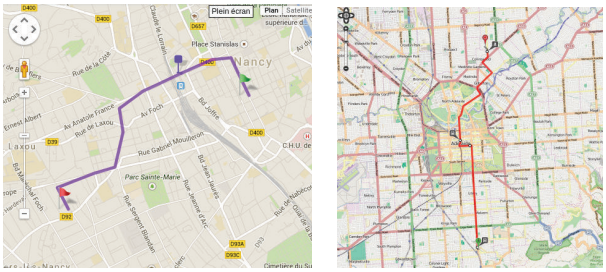

Figure 1. Trip planners of Nancy (left) and Adelaide (right).

Our method relies on two main contributions. First, we consider any individual trip as a set of explicit preferences made by the traveler during the trip. We use this set of pairwise preferences to learn a ranking function of trips.

Proceedings of the 2^{nd} International Workshop on Mining Urban Data, Lille, France, 2015.

This function is then used on the top of the trip planner, to re-rank the k-top recommendations. Second, we model passenger preferences of choosing a specific service or a change point in a way that reflects their dynamic nature. To address the sparse and noisy character of the raw trip, we model the user preferences by a set of dynamic latent variables. We estimate these variables by a smoothed dynamic non-negative factorization of service and transit counts.

The remainder of this paper is organized as follows. In Section 2 we briefly review the state of art in urban trip planning. Section 3 introduces the trip ranking problem by analyzing individual trips for Nancy city case. Learning to rank for trip planning is presented in Section 4. Then Section 5 proposes to model user preferences by dynamic latent variables and develop an estimation method by smoothed dynamic non-negative factorization of service and transit counts. In Section 6, we report results of evaluation on trip re-ranking for two city datasets. Section 7 concludes the paper.

2. Prior Art

Trip planners. Public transport (PT) trip planners are designed to provide information about available journeys in the transport system. The application prompts a user to input an origin o, a destination d and a departure time t_s (or arrival time t_f), it then deploys a trip planning engine to find a sequence of available PT services from o to d starting at time t_s (or ending at time t_f).

Trip planners often retrieve multiple trips for a user query. They typically use a variation of the time-dependent *shortest path algorithm* to search a graph of nodes (representing access points to the network) and edges (representing possible journeys between points) (Casey et al., 2014). Different weightings such as distance, cost or accessibility are often associated with each edge and node. Search may be optimized on different criteria, for example, the fastest, least changes or cheapest ones (Pelletier et al., 2009).

Planning high quality realistic trips remains difficult for several reasons (McGinty & Smyth, 2000). First, available General Transit Feed Specification (GTFS) sources rarely contain all information useful for constructing realistic plans. Second, the notion of "service quality" is difficult to define and is likely to change from person to person. Consequently, in real-world trip planning, the shortest trip is rarely the best one for a given user.

Multiple efforts have been made to improve the trip planning (Lathia & Capra, 2011; Liebig et al., 2014; Mokhtari et al., 2009; Trepanier et al., 2005; Yuan et al., 2011). Analysis of trip planner log files (Trepanier et al., 2005) can help improve transit service by providing better knowledge on transit users. Log files were useful for identifying new locations to be assessed for better understanding user behaviors, and for guiding updates of the PT information system.

Personalization of trip planning took into account user preferences and tries to identify the best trips among a set of possible answers. In (Mokhtari et al., 2009), the fuzzy set theory was used to model complex user preferences. A typology of preferences was proposed to explicitly express the preferences and integrate them in a query language.

Trip personalization by mining public transport data has been addressed in (Lathia & Capra, 2011). It established a relation between urban mobility and fare purchasing habits in London public transport network (Seaborn et al., 2010), and proposed personalized ticket recommendations based on the estimated future travel patterns and matching travelers to the best fare.

Integrating real time information in trip planners has been another research trend. (Yuan et al., 2011) presented a cloud-based system computing customized and practically fast driving routes for an end user using (historical and real-time) traffic conditions and driver behavior. GPS-equipped taxicabs are used as mobile sensors constantly probing the traffic rhythm of a city and taxi drivers' intelligence in choosing driving directions. The real time trip planning has also been extended to multi-modality (Casey et al., 2014; Seaborn et al., 2010). It used data from GPS-enabled vehicles to produce more accurate plans in terms of time and transit vehicles.It incorporates the delays into the transit network at real-time to minimize the gap with respect to the prediction model.

Learning to Rank. In document retrieval, to ranking documents based on their degrees of relevance to a query has been the key question for decades. Much effort has been placed on developing document ranking functions. Early methods used a small number of document features (e.g., term frequency, inversed document frequency, and document length), with an empirical tuning of the ranking function parameters. To avoid the manual tuning, the document retrieval was proposed to be regarded as *learning to rank* (Burges et al., 2005; 2006; Cao et al., 2006; Liu, 2011). Click-through data are used to deduce pair-wise training data for learning ranking functions.

In learning to rank, a number of categories are given and a total order is assumed to exist over the categories. Labeled instances are provided, and each instance is represented by a feature vector, and each label denotes a rank. Existing methods can be categorized as point-wise, pair-wise and list-wise (Liu, 2011). In point-wise methods, each instance with its rank is used as an independent training example. The goal of learning is to correctly map instances into intervals. In pair-wise methods, each instance pair is used as a training example and the goal of training is to correctly

find the differences between ranks of instance pairs, and ranking is transformed into pairwise classification or pairwise regression (Herbrich et al., 2000). This model formalizes learning to rank as learning for classification on pairs of instances and can deploy any classification method. In list-wise methods, the loss function is defined on a ranked list with respect to a query (Xia et al., 2008).

3. Individual trips analysis

We consider a public transportation system that offers a number of services (buses, trams, trains, etc.) to urban travelers. Any individual passenger trip J represents a sequence of PT services and changes between the services. Service legs of $\mathcal{J}$ form a sequence $\mathcal{S}_\mathcal{J} = \{l_1, \ldots, l_n\}, n \geq 1$, where leg l_i is a tuple $(s_i, b_i, a_i, t_i^b, t_i^a)$, s_i is a service identifier (a bus number, for ex.); b_i and a_i are boarding and alighting stops, t_i^b and t_i^a are boarding and alighting timestamps. Trip is *direct* if $n = 1$, and *transit* otherwise.

A transit trip includes $n - 1$ changes which refer to *waiting* and/or *walking* between the services. The sequence of changes is defined as $\mathcal{C}_\mathcal{J} = \{c_1, \ldots, c_{n-1}\}, n \geq 1$, where c_i is uniquely defined by two successive service legs l_i and l_{i+1}, as $c_i = (a_i, b_{i+1}, t_i^a, t_{i+1}^b)$.

We make the following association between individual trips and trip recommendations. We consider a trip $\mathcal{J}$ as an *explicit answer* to an *implicit travel query* $Q = (o = b_1, d = e_n, t_s = t_1^b)$ or $Q = (o = b_1, d = a_n, t_f = t_n^a)$.

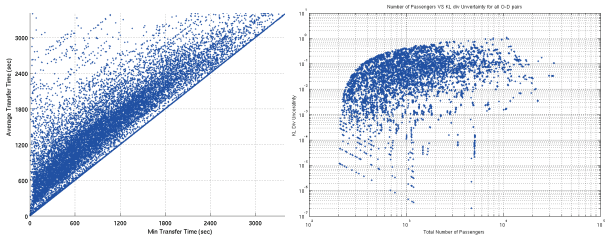

Figure 2. a) Minimal travel time vs average travel time. b) Trip uncertainty.

We analyze sets of individual trips collected from the automated fare collection systems (Mezghani, 2008) installed in Nancy, France and Adelaide, Australia; we mined these data to understand how passengers' choices differ from the planner recommendations.

For every pair of locations (o, d) in a network, we extract all real trips from o to d and analyze their travel time distribution. Figure 2.a shows the distribution of the *minimal* versus the *average* travel time for every (o, d) pair in Nancy. The high density zone suggests that the average travel time is far longer than the minimal time which is conventionally assumed by the planners.

Trip datasets expose a very large variety of paths for any (o,d) pair; the maximum number of different paths observed is 46 for Nancy and 37 for Adelaide; the average number of paths between two locations is 2.71 and 3.12, respectively. We measure the uncertainty of choosing one or another path from an origin o to a destination d, by using the Kullback-Leibler divergence $KL(q||p)$ of the trip distribution q from the uniform distribution p. The higher KL values indicate the higher certainty and a clear domination of one trip over others. Figure 2.b plots the KL divergence values for all (o,d) pairs in Nancy using the log-log scale. Again, the high density zone suggests that a large part of (o,d) pairs is dominated not by one but by 2 to 5 different paths of high frequency.

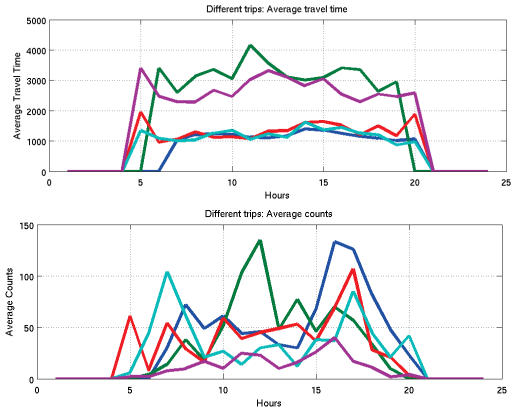

Figure 3. a) 5-top trips for one (origin, destination) pair in Nancy. b) The travel time and trip count distributions for top 5 trips.

It is important to recall that travelling preferences change during the day. Figure 3.a shows 5-top transit trips for an example (o, d) location pair in Nancy. Figure 3.b shows the travel time and average trip counts for 5-top trips for this example. All 5 trips are transit ones with one change. The figure reveals how the user preferences vary during the day. The trip planner recommends the trip shown in *red* for the fastest trip query. First, this recommended trip is not the fastest nor the most frequent one. Second, the trip shown in green is the most frequent during the lunch, despite it is far from being fast.

Figure 4 gives a more general picture. It shows 240 most frequent (o,d) pairs in Nancy. For each pair, Figure 4.a uses the different colors to show changing user preferences. The most frequent trip is colored in dark blue. Second, third, forth and fifth preferences are shown in blue, green, orange and brown colors, respectively. Trips are sorted by the distance between the origin and destination (see Figure 4.b).

Short trips expose a higher variability than longer ones. As the figure shows, the second choices are more visible (blue

color) during the morning rush hours. Figure 4.c shows the trip planner recommendations for the same pairs. The recommendations are static and do not reflect the user preferences.

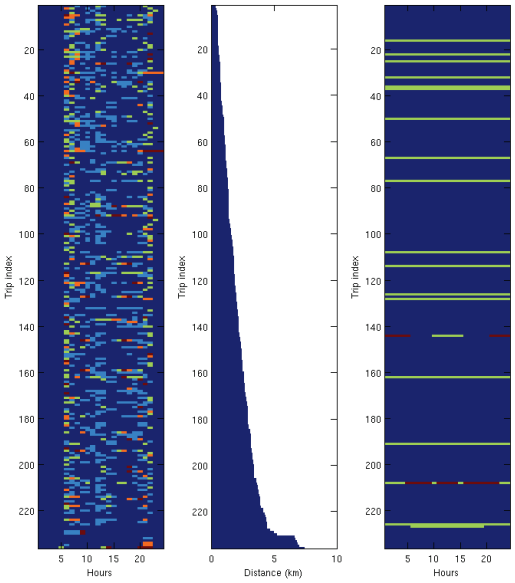

Figure 4. a) Changing user preferences for most frequent (o,d) pairs in Nancy. b) Trip distances. c) Trip recommendations by the planner.

We conclude this section by Figure 5 which shows how the user preferences vary between the PT services. It presents the total passenger counts for all Nancy change points, at 8am, 1pm and 6pm.

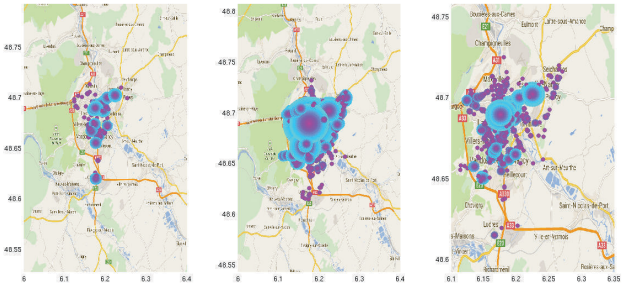

Figure 5. Change counts in Nancy at 8am, 1pm and 6pm.

4. Learning to rank trips

When a passenger travels from an origin o to a destination d at time t_s, she implicitly prefers the trip $\mathcal{J}$ she takes to all other trips $\mathcal{J}', \mathcal{J}' \neq \mathcal{J}$. Our approach is to transform this implicit feedback into an explicit set of pair-wise trip preferences and to learn the ranking function f from them.

Algorithm 1 below uses the trip planner and a set $\mathcal{T}$ of individual passengers' trips. For any trip $\mathcal{J} \in \mathcal{T}$ matching the query $Q = (o, d, t_s)$, the algorithm retrieves the k-top candidates for Q and retains that $\mathcal{J}$ has been preferred to any of these candidates, except $\mathcal{J}$ itself if it happens to be in this set. Real trip $\mathcal{J}$ matches a recommended trip $\mathcal{J}'$, if it has the same number of legs and following the same sequence of services. If $\mathcal{S}_\mathcal{J} = \{l_1, \ldots, l_n\}$ and $S_{\mathcal{J}'} = \{l'_1, \ldots, l'_n\}$, then $\mathcal{J}$ *matches* $\mathcal{J}'$ iff $s_i = s'_i \wedge b_i = b'_i \wedge a_i = a'_i$, for all $i = 1, \ldots, n$.

Algorithm 1 Rank learning algorithm.

Require: Collection $\mathcal{T}$ of passenger trips $\mathcal{J} = (\mathcal{S}, \mathcal{C})$
Require: Trip planner P with k-top recommendations
$S = \emptyset$; `set of pairwise preferences`
for each $\mathcal{J} \in \mathcal{T}$ **do**
  Form a query $Q = (o = b_1, d = a_n, t_s = t^b_1)$
  Query the planner P with query Q
  Retrieve k-top trips as a list L
  for each $\mathcal{J}' \in L, \mathcal{J}' \neq \mathcal{J}$ **do**
    Add $(Q, \mathbf{x}(\mathcal{J}) \succ \mathbf{x}(\mathcal{J}'))$ to S
  end for
end for
Learn the ranking model f from S
Ensure: f

Once the ranking function f is learned, it can be used to improve the relevance of trip planner recommendations acording to the *re-ranking scenario*. The trip planner does not change the way it works. And for a new user query Q, the trip planner first generates k-top candidate trips. Then these candidates are re-ranking using the function f.

To learn a ranking function f, Algorithm 1 requires every trip $\mathcal{J}$ be described by a feature vector $\mathbf{x}(\mathcal{J})$. In the following sections, we first describe a method for learning the ranking function f and then how to extract relevant and dynamic features from individual trips.

4.1. Gradient Boosting Rank

We used individual trips to form a set pairwise preferences, a ranking function f can be learned from. For each individual trip $\mathcal{J} \in \mathcal{T}$, we generate a set of labeled data $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,m_i}, y_{i,m_i})$, $i = 1, \ldots, |\mathcal{T}|$, which are preference pairs of feature vectors. If $\mathbf{x}_{i,j}$ has a higher rank than $\mathbf{x}_{i,k}$ ($y_{i,j} > y_{i,k}$), then $\mathbf{x}_{i,j} \succ \mathbf{x}_{i,k}$ is a preference pair, which means that $\mathbf{x}_{i,j}$ is ahead of $\mathbf{x}_{i,k}$. The preference pairs can be viewed as instances and labels in a new classification problem, where $\mathbf{x}_{i,j} \succ \mathbf{x}_{i,k}$ is a positive instance.

Any classification method can be used to train a classifier $f(\mathbf{x})$ which is then used for ranking. Trips are assigned scores by $f(\mathbf{x})$ and sorted by the scores. Learning a good

ranking model is realized by training of a model for pairwise classification. The loss function in learning is pairwise because it is defined on a pair of feature vectors.

The pairwise approach is adopted in many methods, including Ranking SVM (Herbrich et al., 2000), RankBoost (Freund et al., 2003), RankNet (Burges et al., 2005), IR SVM (Tsai et al., 2007), GBRank (Zheng et al., 2007), LambdaRank (Burges et al., 2006), and others. In the following we adopt GBRank as one of popular pairwise methods currently used.

GBRank takes preference pairs as training data, $\{\mathbf{x}_i^1, \mathbf{x}_i^2\}, \mathbf{x}_i^1 \succ \mathbf{x}_i^2, i = 1, \ldots, N.$ and uses the parametric pairwise loss function

$$L(f) = \frac{1}{2}\sum_{i=1}^{N}(\max\{0, \tau - (f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2)\})^2,$$

where $f(\mathbf{x})$ is the ranking function and τ is a parameter, $0 < \tau \leq 1$. The loss is 0 if $f(\mathbf{x}_i^1)$ is larger than $f(\mathbf{x}_i^2) + \tau$, otherwise, the incurred loss is $\frac{1}{2}(f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau)^2$.

To optimize the loss function with respect to the training instances, the Functional Gradient Decent is deployed. Treating all $f(\mathbf{x}_i^1), f(\mathbf{x}_i^2), i = 1, \ldots, N$ as variables; the gradient of $L(f)$ is computed with respect to the training instances as follows

$$-\max\{0, f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau\}, \max\{0, f(\mathbf{x}_i^2) - f(\mathbf{x}_i^1) + \tau\}$$
$$i = 1, \ldots, N.$$

If $f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2) \geq \tau$, the corresponding loss is zero, and there is no need to change the ranking function. If $f(\mathbf{x}_i^1) - f(\mathbf{x}_i^2) < \tau$, the loss is non-zero, and the ranking function is updated using the Gradient Descent:

$$f_k(\mathbf{x}) = f_{k-1}(\mathbf{x}) - \nu\Delta L(f_k(\mathbf{x})),$$

where $f_k(\mathbf{x})$ and $f_{k-1}(\mathbf{x})$ denote the values of $f(\mathbf{x})$ at k-th and $(k-1)$-th iterations, respectively, ν is the learning rate.

At the k-th iteration of the learning, GBRank collects all the pairs with non-zero losses $\{(\mathbf{x}_i^1, f_{k-1}(\mathbf{x}_1^2) + \tau), (\mathbf{x}_i^2, f_{k-1}(\mathbf{x}_i^1) - \tau)\}$ and employs Gradient Boosting Tree (Friedman, 2000) to learn a regression model $g_k(\mathbf{x})$ that can make prediction on the regression data. The learned model $g_k(\mathbf{x})$ is then linearly combined with the existing model $f_{k-1}(\mathbf{x})$ to create a new model $f_k(\mathbf{x})$ as follows

$$f_k(\mathbf{x}) = \frac{kf_{k-1}(\mathbf{x}) + \beta_k g_k(\mathbf{x})}{k+1},$$

with β_k as a shrinkage factor (Zheng et al., 2007).

5. Trip feature extraction

We now describe each real trip $\mathcal{J}$ by a set of relevant and dynamic features $\mathbf{x}(\mathcal{J})$. There may exist explicit and implicit factors which influence the passenger choice. Passengers make their choices in the function of location and time.

We mention two groups of trip features. First, **global features** describe the whole trip; they are the travel time, the number of changes, the usage of specific types of transport (bus, train, tram, etc.), multi-modality, etc. Second, much more relevant and specific are **local features** that describe each service leg and change that compose a given trip. For each PT service, we may extract the estimated means and variance of the speed when using this line at this time period, the average delay with respect to the schedule. For each change point, we can estimate the walking distance if any, the closeness to a commercial zone or transportation hub, etc.

Unfortunately, raw features of services and change counts are generally sparse, noisy and prone to many errors. Main reasons for errors are due to incorrect setup of ticket validation machines, lack of alignment between ticket validation machines and GPS localization, and card misuse by travelers.

So we intend to extract such latent features from sparse and noisy counts that be able to represent user preferences and their dynamic character.

We split all trips $\mathcal{J} \in \mathcal{T}$ in two collections of service and change observations, $\mathcal{A}^s = \{l_i | l_i \in S_\mathcal{J}, \mathcal{J} \in \mathcal{T}\}$ and $\mathcal{A}^c = \{c_i | c_i \in C_\mathcal{J}, \mathcal{J} \in \mathcal{T}\}$. In the following we assume for brevity working with a set of observations $\mathcal{A}$; it may indicate service or change observations, or their sum.

If we split all observations in $\mathcal{A}$ in T time periods, so we obtain a sequence of count matrices $\mathbf{A}_t, t = 1, \ldots, T, \mathbf{A}_t \in R_+^{p\times p}$ at time period t, where a_{ij} is the service or change count during the period t. and p is the number of stops.

The full diagram of latent feature extraction for individual trips and learning the ranking function is given in Figure 6.

5.1. Collapsed matrices

We first consider the static case when T is 1 and all observations from $\mathcal{A}$ are collapsed in one matrix $\mathbf{A}$.

Both service and change data are sparse non-negative counts, and we can use the non-negative matrix factorization (NNMF) as a method giving a great low-rank robust interpretation of data (Lee & Seung, 2001). They can be efficiently computed by formulating the penalized optimization problem and using modern gradient-descent algorithms (Hoyer, 2004).

Matrix A is approximated with a product ot two low-rank matrices that is estimated through the following minimiza-

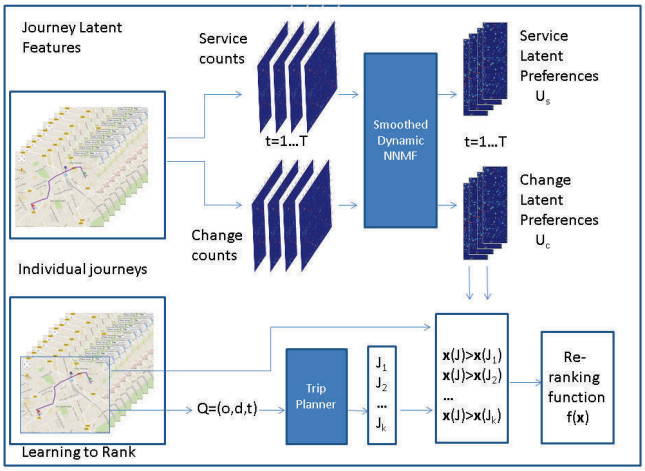

Figure 6. Preference features and re-ranking function learning.

tion

$$\min_{\mathbf{U}\geq 0,\mathbf{V}\geq 0}||\mathbf{A}-\mathbf{U}\mathbf{V}^T||_F^2,$$

where $\mathbf{U}$ and $\mathbf{V}$ are $n \times K$ non-negative matrices. The rank or dimension of the approximation K corresponds to the number of latent factors; it is chosen to obtain a good data fit and interpretability, where $\mathbf{U}$ give latent factors for origin stops and $\mathbf{V}$ does for destination stops.

The factorized matrices are obtained by minimizing an objective function that consists of a goodness of fit term and a roughness penalty

$$\min_{U\geq 0,V\geq 0}||\mathbf{A}-\mathbf{U}\mathbf{V}^T||_F^2+\lambda(||\mathbf{U}||_1+||\mathbf{V}||_1), \quad (1)$$

where the parameter $\lambda \geq 0$ indicates the penalty strength; a larger penalty encourages sparser matrices $\mathbf{U}$ and $\mathbf{V}$. Adding penalties to NMF is a common strategy since they not only improve interpretability, but often improve numerical stability of the estimation.

5.2. Smoothed Dynamic NNMF

In the general case $T > 1$, we have a sequence of matrices $\{\mathbf{A}_t\}_{t=1}^T$ for time periods $t = 1,\ldots,T$. To produce a sequence of low-rank matrix factorizations $\{\mathbf{U}_t,\mathbf{V}_t\}_{t=1}^T$, we can extend the factorization in (1) to the case $T > 1$ by independent factorization of T matrices $\{\mathbf{A}_t\}$. However, we additionally impose a smoothness constraint on both $\mathbf{U}_t$ and $\mathbf{V}_t$, in order to force the latent factors to be similar to the previous time periods, in both boardings and alightings. The objective function then becomes

$$\begin{aligned}&\min_{\mathbf{U}_t\geq 0,\mathbf{V}_t\geq 0}||\mathbf{A}_t-\mathbf{U}_t\mathbf{V}_t^T||_F^2\\&+\mu\textstyle\sum_{t=2}^T(||\mathbf{U}_t-\mathbf{U}_{t-1}||_2^F+||\mathbf{V}_t-\mathbf{V}_{t-1}||_2^F)\\&+\lambda(\textstyle\sum_{t=1}^T||\mathbf{U}_t||_1+||\mathbf{V}_t||_1),\end{aligned} \quad (2)$$

where parameters λ,μ are set by the user. The objective function imposes smoothing $\mathbf{U}_t$ and $\mathbf{V}_t$ on two successive time periods, but it can be generalized to a larger window.

To estimate matrices $\mathbf{U}_t$ and $\mathbf{V}_t$, we use an extended version of the multiplicative updating algorithm for NNMF (Gillis & Glineur, 2012; Lee & Seung, 2001; Mankad & Michailidis, 2013), based an adaptive gradient descent.

Temporal extensions of matrix factorization techniques have been studied in (Elsas & Dumais, 2010; Mankad & Michailidis, 2013; Saha & Sindhwani, 2012; Sun et al., 2014). (Elsas & Dumais, 2010) analyzed the temporal dynamics of Web document content. To improve the relevance ranking, it developed a probabilistic document ranking algorithm that allows differential weighting of terms based on their temporal characteristics. (Sun et al., 2014) addressed recommendation systems with significant temporal dynamics; it developed the collaborative Kalman filter which extends probabilistic matrix factorization in time through a state-space model. Community detection in time-evolving graphs is analyzed in (Mankad & Michailidis, 2013). The latent structure of overlapping communities is discovered through the sequential matrix factorization.

To solve (2), we follow (Mankad & Michailidis, 2013) and consider the Lagrangian as follows

$$\begin{aligned}&L=||\mathbf{A}_t-\mathbf{U}_t\mathbf{V}_t^T||_F^2+\\&+\mu\textstyle\sum_{t=2}^T(||\mathbf{U}_t-\mathbf{U}_{t-1}||_2^F+||\mathbf{V}_t-\mathbf{V}_{t-1}||_2^F)\\&+\textstyle\sum_{t=1}^T(\lambda(||\mathbf{U}_t||_1+||\mathbf{V}_t||_1)+Tr(\Phi\mathbf{U}_t)+Tr(\Psi\mathbf{V}_t)),\end{aligned} \quad (3)$$

where Φ,Ψ are Lagrange multipliers. The method works as an adaptive gradient descent converging to a local minimum. Kuhn-Tucker (KKT) optimality guarantees the necessary conditions for convergence [44]. The KKT optimality conditions are obtained by setting $\frac{\partial L}{\partial \mathbf{U}_t}=0;\frac{\partial L}{\partial \mathbf{V}_t}=0, t=1,\ldots,T$. It can be shown that the KKT optimality conditions are obtained by

$$\begin{aligned}\Phi_t&=-2\mathbf{A}_t\mathbf{V}_t+2\mathbf{U}_t\mathbf{V}_t^T\mathbf{V}_t-2\mu(\mathbf{U}_{t-1}-\mathbf{U}_t)+2\lambda,\\\Psi_t&=-2\mathbf{A}_t^T\mathbf{U}_t+2\mathbf{V}_t\mathbf{U}_t^T\mathbf{U}_t-2\mu(\mathbf{V}_{t-1}-\mathbf{V}_t)+2\lambda,\end{aligned} \quad (4)$$

which after matrix algebra manipulations lead to the multiplicative updating rules presented in Algorithm 2.

The convergence of the multiplicative updating algorithm is often reported slow. In practice we obtain meaningful factorizations after a handful of iterations, which we tend to explain by the sparseness of input matrices A_t. In the future, when working with the dense data, faster methods like active set version of the alternating non-negative least squares (ANLS) algorithm (Kim & Park, 2008) will be more appropriate.

Algorithm 2 Dynamic Smoothing NNMF algorithm.

Require: Matrices $\mathbf{A}_t, t = 1, \ldots, T$, constants λ ,μ
Initialize $\mathbf{U}_t, \mathbf{V}_t$ as dense, positive random matrices
repeat
  for t = 1,. . . ,T **do**
    $\mathbf{U}_t \leftarrow \mathbf{U}_t(\mathbf{U}_t\mathbf{V}_t^T\mathbf{V}_t + \lambda\mathbf{A}\mathbf{U}_t)^{-1}(\mathbf{A}_t\mathbf{V}_t + \mu\mathbf{U}_{t-1})$
    $\mathbf{V}_t \leftarrow \mathbf{V}_t(\mathbf{V}_t\mathbf{U}_t^T\mathbf{U}_t + \lambda\mathbf{A}\mathbf{V}_t)^{-1}(\mathbf{A}_t^T\mathbf{U}_t + \mu\mathbf{V}_{t-1})$
  end for
until Convergence
Ensure: $\mathbf{U}_t, \mathbf{V}_t, t = 1, \ldots, T$

5.3. Dynamic trip features

Algorithm 2 finds sparse factorized matrices for a sequence of input matrices $\mathbf{A}_t, t = 1, \ldots, T$. We first apply the algorithm to sequences of service matrices $\mathbf{A}_t^s$ and change matrices $\mathbf{A}_t^c$, extracted from the full trip collection. We thus obtain smoothed factorized matrices $\mathbf{U}_t^s, \mathbf{V}_t^s$, and $\mathbf{U}_t^c, \mathbf{V}_t^c, t = 1, \ldots, T$ for services and changes, respectively. At time period t, a boarding stop b has latent factors given by a corresponding row in $\mathbf{U}_t^s$ this row is denoted $\mathbf{U}_t^s(b)$. For an alighting stop a, row $\mathbf{V}_t^s(a)$ gives the latent factors at time t. We then apply the algorithm to the sum matrices, $\mathbf{A}_t^f = \mathbf{A}_t^c + \mathbf{A}_t^s, t = 1, \ldots, T$. The smoothed factorized matrices for $\mathbf{A}_t^f$ are denoted $\mathbf{U}_t^f, \mathbf{V}_t^f$.

To generate a feature vector $\mathbf{x}$ for a trip $\mathcal{J}$, we may use its decomposition into service legs and changes, $\mathcal{J} = (\mathcal{S}, \mathcal{C})$. The vector $\mathbf{x}(\mathcal{J})$ is then composed of a general feature vector $\mathbf{x}_g$ and four latent components, $\mathbf{x}(\mathcal{J}) = \{\mathbf{x}_g, \mathbf{x}_b^s, \mathbf{x}_a^s, \mathbf{x}_b^c, \mathbf{x}_b^c\}$, where

- $\mathbf{x}_b^s, \mathbf{x}_a^s$ are latent feature vectors averaged over the trip boarding and alighting places, respectively,

$$\mathbf{x}_b^s = \frac{1}{n}\sum_{i=1}^{n}\mathbf{U}_{t_i^b}^s(b_i); \mathbf{x}_a^s = \frac{1}{n}\sum_{i=1}^{n}\mathbf{V}_{t_i^a}^s(a_i);$$

- $\mathbf{x}_b^c$, $\mathbf{x}_a^c$ are latent feature vectors averaged over the change places (alighting and boarding), respectively,

$$\mathbf{x}_b^c = \frac{1}{n-1}\sum_{i=1}^{n-1}\mathbf{U}_{t_i^b}^c(b_i); \mathbf{x}_a^c = \frac{1}{n-1}\sum_{i=1}^{n-1}\mathbf{V}_{t_i^a}^c(a_i).$$

In the case of sum latent matrices $\mathbf{U}_t^f, \mathbf{V}_t^f$, $\mathbf{x}(\mathcal{J})$ is composed of a general feature vector $\mathbf{x}_g$ and two latent components, $\mathbf{x}(\mathcal{J}) = \{\mathbf{x}_g, \mathbf{x}_b^f, \mathbf{x}_a^f\}$ obtained from $\mathbf{U}_t^f$ and $\mathbf{V}_t^f$.

6. Evaluation

To test our method for learning a ranking function from individual trips, we processed 5.2M individual trips collected in Nancy, France during 3 months in 2012. Nancy PT network includes 1129 nodes/stops and offers 107 bus and tram services to travelers. We also processed 12.5M trips from Adelaide, Australia collected during 2.5 months in 2013. Adelaide network offers 312 bus and tram service variations, and accounts for 3524 stops.

To evaluate the impact of modeling user preferences from actual trips, we selected 240 most frequent origin-destination pairs in Nancy (see Figure 4) and 160 most frequent pairs in Adelaide.

When generating temporal sequences of count matrices, we test two cases of $T = 24$ and T=48, when any matrix includes all passenger counts during one hour or 30 minutes. Once a matrix sequence is generated, any matrix is randomly split into 70% for training data and the remaining 30% for testing. All results below are means and variances over 10 independent runs.

We retrieved the trip planner recommendations for Nancy[1] and Adelaide[2]. We learn the ranking function and use it to re-rank the trip recommendations, using different options described in previous sections. To understand the effect of raw count factorization, we consider several options. First, we collapse matrices so disregarding the temporal aspect. Second, we consider either the service $\mathbf{A}_t^s$ and change matrices $\mathbf{A}_t^c$ separately, or sum them up $\mathbf{A}_t^f = \mathbf{A}_t^s + \mathbf{A}_t^c$ before the factorization. Third, we study the effect of temporal smoothing, when factorization is done either independent or by smoothing over successive time periods. Finally, we test different values K for the factorization.

In all experiments with GBRank (see Section 4.1), parameter τ was set to $\tau = 0.3$ and shrinkage factors β_k to 0.8. For smoothed dynamic NNMF, optimal values of μ and λ have been determined by cross-validation. For evaluating the results of ranking methods, we use a measure commonly used in information retrieval, Normalized Discounted Cumulative Gain (NDCG). We choose the perfect ranking's NDCG score 1 which is the error rate of the 1-top recommendation.

Table 6 reports the evaluation results for 12 different methods and compares them to the trip planner baseline for both cities. The analysis of these results provide some interesting insights. First, results are globally better for smaller Nancy than for bigger Adelaide, for both $T = 24$ and $T = 48$ cases. Second, collapsed matrices improve the baseline somewhat, but only taking into account temporal user preferences does really boost the performance. Moreover, smoothed matrix factorization improves considerably over the independent one. Third, the change latent variables appear to be more relevant than services ones. In-

[1] http://www.reseau-stan.com/
[2] https://www.adelaidemetro.com.au/

City	Nancy		Adelaide	
Method	$T = 24$	$T = 48$	$T = 24$	$T = 48$
Baseline: Trip Planner	24.91 ± 1.20	24.91 ± 1.28	38.17 ± 2.28	38.17 ± 2.28
Collapsed:Services	24.73 ± 1.17	24.73 ± 1.22	29.97 ± 2.11	29.97 ± 2.11
Collapsed:Changes	19.69 ± 1.01	19.69 ±1.09	28.63 ± 2.29	28.63 ± 2.29
Collapsed:Services+Changes	19.30 ± 1.14	19.30 ± 1.03	28.05 ± 2.32	28.05 ± 2.32
Collapsed:Sum	19.59 ± 1.13	19.59 ± 1.10	28.17 ± 2.18	28.17 ± 2.18
Indep: Services	14.08 ± 0.92	15.33 ± 0.97	25.33 ± 2.07	24.87 ± 1.87
Indep: Changes	9.55 ± 0.90	9.89 ± 0.86	23.89 ± 1.67	23.93 ± 1.75
Indep: Services+Changes	9.52 ± 0.89	9.41 ± 0.87	22.41 ± 1.72	22.15 ± 1.55
Indep: Sum	10.42 ± 0.89	9.37 ± 0.86	22.37 ± 1.56	23.55 ± 1.59
Smooth: Services	9.22 ±**0.77**	9.37 ± 0.78	15.37 ± 1.38	14.71 ± 1.24
Smooth: Changes	6.71 ± 0.82	6.69 ± **0.74**	16.69 ± **1.24**	16.69 ±1.15
Smooth: Services+Changes	**5.83** ±0.81	**6.12**± 0.79	**14.12**±1.29	**13.63**±**1.14**
Smooth: Sum	7.63 ± 0.79	7.05 ± 0.81	15.05 ± 1.41	14.43 ± 1.32

Table 1. NDCG@1 values for 12 methods and two cities.

stead, using sum counts performs worse than keeping service and change variables separately. We tend to explain this by heterogeneity of service and change preferences.

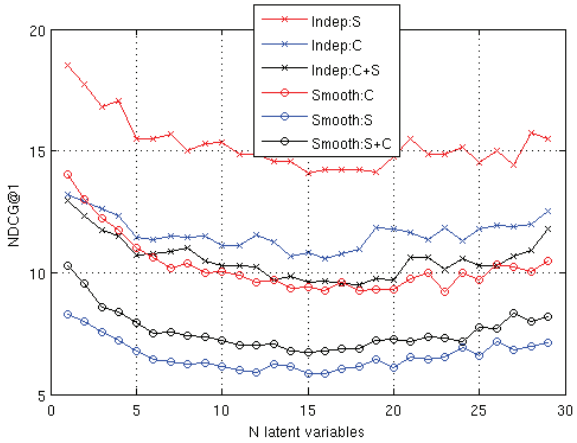

Figure 7. Independent an smoothed predictions vs Number of latent variables.

Figure 7 shows the performance of 3 independent and 3 smoothed methods for $T = 24$ for Nancy, with the number of latent variables K varying between 2 and 30. Surprisingly, already K=2 performs well enough, thus indicating the sparsity of the count matrices.

Figure 8 reports the hour-per-hour performance for the same six methods for Nancy case. Rush hours and lunch time appear to be hard for all methods; the error is the smallest for the periods 10am-12am and 2pm-4pm that points to the correlation between the traffic and trip variability. The traffic growth pushes travelers away from the conventional traveling choices.

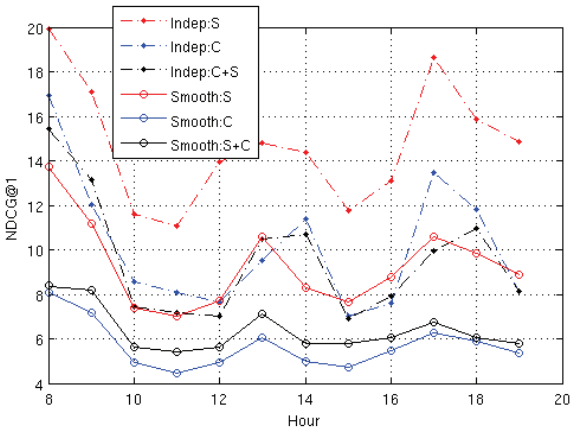

Figure 8. NDCG@1: Independent and smoothed predictions during the day.

7. Conclusion

We address the problem of relevance of trips recommended by urban trip planners. We analyzed passengers' trips extracted from two public transportation systems. We propose a method for improving the recommendation relevance by learning from choices made by travelers who use the transportation system daily. We convert the actual trips into a set of pairwise preferences and learn a ranking function using the Gradient Boosting Rank method. We describe actual trips with a number of time-dependent latent features, and develop a smoothed non-negative matrix factorization to estimate the latent variables of user preferences while choosing PT services and change points. Experiments with real trip data demonstrate that the re-ranked trips are measurably closer to those actually chosen by passengers than are the trips produced by planners with static

heuristics.

References

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML'05*, pages 89–96, 2005.

C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Proc. NIPS'06*, pages 193–200, 2006.

Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proc. SIGIR '06*, pages 186–193, New York, NY, USA, 2006. ACM.

B. Casey, A. Bhaskar, H. Guo, and E. Chung. Critical review of time-dependent shortest path algorithms: A multimodal trip planner perspective. *Transport Reviews*, 34:522–539, 2014.

J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proc. WSDM '10*, pages 1–10, New York, NY, USA, 2010. ACM.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Machine Learning Res.*, 4:933–969, 2003.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

N. Gillis and F. Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural Comput.*, 24(4):1085–1105, April 2012.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, 2000.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, December 2004.

H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl.*, 30(2):713–730, July 2008.

N. Lathia and L. Capra. Mining mobility data to minimise travellers' spending on public transport. In *Proc. ACM KDD'11*, pages 1181–1189, 2011.

D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. NIPS'01*, pages 556–562, 2001.

J. Letchner, J. Krumm, and E. Horvitz. Trip router with individualized preferences: Incorporating personalization into route planning. In *Proc. IAAI'06 - Vol 2*, pages 1795–1800. AAAI Press, 2006.

T. Liebig, N. Piatkowski, C. Bockermann, and K. Morik. Predictive trip planning-smart routing in smart cities. In *EDBT/ICDT Workshops*, pages 331–338, 2014.

T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

S. Mankad and G. Michailidis. Structural and functional discovery in dynamic networks with non-negative matrix factorization. *Phys. Rev. E*, 88:042812, Oct 2013.

L. McGinty and B. Smyth. Turas: A personalised route planning system. In *Proc. PRICAI'00*, pages 791–791, Berlin, Heidelberg, 2000. Springer-Verlag.

M. Mezghani. Study on electronic ticketing in public transport. *European Metropolitan Transport Authorities (EMTA)*, 38:1–56, 2008.

A. Mokhtari, O. Pivert, and A. HadjAli. Integrating complex user preferences into a route planner: A fuzzy-set-based approach. In *IFSA/EUSFLAT Conf.*, pages 501–506, 2009.

M.-P. Pelletier, M. Trepanier, and C. Morency. Smart card data in public transit planning: A review. *CIRRELT Rapport 2009-46*, November 2009.

A. Saha and V. Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *Proc. WSDM'12*, pages 693–702, 2012.

C. Seaborn, J. Attanucci, and N. H. M. Wilson. Analyzing multimodal public transport journeys in london with smart card fare payment data. *Transportation Research Record: J. Transp. Research Board*, 2121:55–62, 2009.

J. Z. Sun, D. Parthasarathy, and K.R. Varshney. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Trans. on Signal Processing*, 62(14):3499–3509, July 2014.

M. Trepanier, R. Chapleau, and B. Allard. Can trip planner log files analysis help in transit service planning? *Journal of Public Transportation*, 8(2):79–103, 2005.

M.-F. Tsai, Tie-Yan Liu, T. Qin, H.-H Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proc. SIGIR'07*, pages 383–390, 2007.

F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. ICML'08*, pages 1192–1199, 2008.

J. Yuan, Yu Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *KDD '11*, pages 316–324, New York, NY, USA, 2011. ACM.

Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proc. SIGIR '07*, pages 287–294, New York, NY, USA, 2007. ACM.