RecTour 2017

2nd Workshop on Recommenders in Tourism

Como, Italy, August 27th, 2017

Proceedings

Edided by Julia Neidhardt, Daniel Fesenmaier,
Tsvi Kuflik, and Wolfgang Wörndl

Co-located with the 11th ACM Conference on Recommender Systems (RecSys 2017)

Copyright and Bibliographical Information

This volume is published by Julia Neidhardt, Daniel Fesenmaier, Tsvi Kuflik and Wolfgang Wörndl.
Published online at http://ceur-ws.org/Vol-1906/ (ISSN 1613-0073).

Proceedings of the 2nd Workshop on Recommenders in Tourism (RecTour 2017), held in conjunction with the 11th ACM Conference on Recommender Systems (RecSys 2017), August 27th - 31st, 2017, Como, Italy, https://recsys.acm.org/recsys17/.

Julia Neidhardt, Daniel Fesenmaier, Tsvi Kuflik and Wolfgang Wörndl (editors).

Further information about the workshop can be found at: http://www.ec.tuwien.ac.at/rectour2017/

Preface

This volume contains the contributions presented at the Workshop on Recommenders in Tourism (RecTour), hold in conjunction with the 11th ACM Conference on Recommender System (RecSys 2017), in Como, Italy. The proceedings are also published online by CEUR Workshop Proceedings at http://ceur-ws.org/Vol-1906/.

RecTour 2017 focuses on a variety of challenges specific to recommender systems within the tourism domain. In this domain, there are considerably more complicated scenarios than finding the best product for a user. Planning a vacation usually involves searching for a set of products that are interconnected (e.g. means of transportation, lodging, attractions), with a rather limited availability, and where contextual aspects may have a major impact (e.g. spatiotemporal context, social context). In addition and most importantly, products are emotionally "loaded" and experientially based; therefore, decision taking is not based solely on rational and objective criteria (i.e. system 2 thinking). As such, providing the right information to visitors of a tourism site at the right time about the site itself and various services nearby is challenging. Additionally, and in contrast to many other domains, information providers are normally small - medium enterprises (SMEs) that do not have full information about available opportunities. Moreover, there is no single, standard format to house this information. Last, much of the tourism experience is co-produced; that is, it occurs during the consumption of the product and therefore, the context of the recommendation is extremely important. Thus, given this diversity, building effective recommenders within the tourism domain is extremely challenging.

The rapid development of information and communication technologies (ICT) in general and the Web has transformed the tourism domain whereby most travelers rely little on travel agents or agencies. Indeed, recent studies indicate that travelers are now active in searching for information and composing their vacation packages according to their specific preferences. When onsite, they search for freely available information about the site itself rather than renting a visitor guide that may be available, but considered to be expensive and sometimes outdated. However, like in many other cases, the blessing of the web comes with a curse, the curse of information overload. Recommender systems have been suggested as a practical tool for overcoming this overload. However, the tourism domain is substantially more complicated, and as such, creates huge challenges for those designing tourism-focused recommenders.

This workshop aims at bringing together researchers and practitioners working in the tourism recommendation domain in order to look at the challenges from the point of view of the user interactions as well as from the perspective of service providers as well as from additional stakeholders (e.g. destination management organizations). Further, the workshop aims at attracting presentations of novel ideas for addressing these challenges with the goal to advance the current state of the art in this field by providing a forum for researchers and practitioners from different fields, e.g., tourism, recommender systems, user modelling, user interaction, mobile, ubiquitous and ambient technologies, artificial intelligence and web information systems, to explore various practical use cases of applications of these technologies in tourist recommenders of the future. The overall goal is to identify and discuss in depth various user groups, tasks and roles needed to achieve personalization, as to further enhance recommendations for tourism applications. RecTour 2017 aims to continue the community building process and the discussions started at RecTour 2016.

August 2017 Julia Neidhardt, Daniel Fesenmaier, Tsvi Kuflik and Wolfgang Wörndl

Workshop Committees

Organizers

- Julia Neidhardt, TU Wien, Austria
- Daniel Fesenmaier, University of Florida, USA
- Tsvi Kuflik, The University of Haifa, Israel
- Wolfgang Wörndl, TU München, Germany

Program Committee

- Derek Bridge, University College Cork, Ireland
- Damianos Gavalas, University of the Aegean, Greece
- Ulrike Gretzel, The University of Queensland, Australia
- Wilfried Grossmann, University of Vienna, Austria
- Dietmar Jannach, TU Dortmund, Germany
- Antonio Moreno, Universitat Rovira i Virgili, Spain
- Francesco Ricci, University of Bozen/Bolzano, Italy
- Hannes Werthner, TU Wien, Austria
- Zheng Xiang, Virginia Tech, USA
- Markus Zanker, University of Bozen/Bolzano, Italy

Workshop Program

9:00 - 10:30 Session 1: Opening and Keynote Presentation

• Workshop opening
• Keynote by Neal Lathia (Skyscanner, UK)

10:30 - 11:00 Coffee break

11:00 - 12:30 Session 2: Long Paper Presentations

• Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy and Elena Baralis: *Predicting Your Next Stop-over from Location-based Social Network Data with Recurrent Neural Networks.*
• Victor A. Arrascue Ayala, Kemal Cagin Gülsen, Anas Alzogbi, Michael Färber, Marco Muñiz and Georg Lausen: *A Delay-Robust Touristic Plan Recommendation Using Real-World Public Transportation Information.*
• Christopher Laß, Daniel Herzog and Wolfgang Wörndl: *Context-Aware Tourist Trip Recommendations.*

12:30 - 14:00 Lunch break

14:00 - 15:30 Session 3: Short and Position Paper Presentations

• Gunjan Kumar, Houssem Jerbi and Michael O'Mahony: *Towards the Recommendation of Personalised Activity Sequences in the Tourism Domain.*
• Diana Nurbakova, Léa Laporte, Sylvie Calabretto and Jerome Gensel: *Itinerary Recommendation for Cruises: User Study.*
• Tom Gross: Group Recommender Systems in Tourism: *From Predictions to Decisions.*
• Catalin-Mihai Barbu and Jürgen Ziegler: *Co-Staying: A Social Network for Increasing the Trustworthiness of Hotel Recommendations.*

15:30 - 16:00 Coffee break

16:00 - 17:30 Session 4 : Panel Discussion and Workshop Closing

• Panel discussion: *Specific challenges for tourism recommender systems seen from academic and the industry perspectives.*
- Neal Lathia (Skyscanner, UK)
- Markus Zanker (University of Bozen/Bolzano, Italy)
- David Zibriczky (trivago, Germany)
• Workshop closing

Table of Contents

Predicting Your Next Stop-over from Location-based Social Network Data with Recurrent Neural Networks

Enrico Palumbo
ISMB
Turin, Italy
EURECOM
Sophia Antipolis, France
palumbo@ismb.it

Giuseppe Rizzo
ISMB
Turin, Italy
giuseppe.rizzo@ismb.it

Raphaël Troncy
EURECOM
Sophia Antipolis, France
raphael.troncy@eurecom.fr

Elena Baralis
Politecnico di Torino
Torino, Italy
elena.baralis@polito.it

ABSTRACT

In the past years, Location-based Social Network (LBSN) data have strongly fostered a data-driven approach to the recommendation of Points of Interest (POIs) in the tourism domain. However, an important aspect that is often not taken into account by current approaches is the temporal correlations among POI categories in tourist paths. In this work, we collect data from Foursquare, we extract timed paths of POI categories from sequences of temporally neighboring check-ins and we use a Recurrent Neural Network (RNN) to learn to generate new paths by training it to predict observed paths. As a further step, we cluster the data considering users' demographics and learn separate models for each category of users. The evaluation shows the effectiveness of the proposed approach in predicting paths in terms of model perplexity on the test set.

KEYWORDS

Sequence learning, path recommendation, tourism, POI recommendation

1 INTRODUCTION

Location-based Social Networks (LBSN) allow users to *check-in* in a Point-of-Interest (POI)[1] and share their activities with friends, providing publicly available data about their behavior. One of the distinctive features of LBSN data with respect to traditional location prediction systems, which are mainly based on GPS data and focus on physical mobility [33], is the rich categorization of POIs in consistent taxonomies, which attribute an explicit semantic meaning to users' activities. The availability of venue categories has opened new research lines, such as statistical studies of venues peculiarities [17], automatic creation of representations of city neighborhoods and users [22, 25], definition of semantic similarities between cities [24]. Most importantly, venue categories play an important role in POI recommender systems, as they enable to model user interests and personalize the recommendations [18]. In the past years, little attention has been dedicated to the temporal correlations among venue categories in the exploration of a city, which is nonetheless a crucial factor in recommending POIs. Consider the example of a check-in in an *Irish Pub* at 8 PM: is the user more likely to continue her evening in a *Karaoke Bar* or in an *Opera House*? Better a *Chinese Restaurant* or an *Italian Restaurant* for dinner after a *City Park* in the morning and a *History Museum* in the afternoon? Note that predicting these sequences require an implicit modeling of at least two dimensions: 1) temporal, as certain types of venues are more temporally related than others (e.g. after an *Irish Pub*, people are more likely to go to *Karaoke* than to a *History Museum* 2) personal, as venue categories implicitly define a user profile, independently from their order (e.g. *Steakhouse* and *Vegetarian Restaurant* do not go frequently together). Most of existing studies attempt to model directly sequences of POIs rather than their categories to recommend the next POI to a user (see 'next POI prediction' in Sec. 2). In this work, we focus on modeling sequences of POI categories to enhance the generality and the portability of the obtained results. This can be considered as a first step in the next POI prediction problem, as the POI category can then be turned into a specific POI by querying a database of POIs according to a variety of parameters, such as the user context (e.g. position, weather) and/or specific POI features such as popularity, average prices and the like.

In order to address this problem, we first collect users' check-ins from Foursquare and extract their corresponding venue categories, segmenting them into a set of temporally neighboring activities, which we call *paths*. Then, we train a Recurrent Neural Network to learn to predict these paths in order to generate new ones, thanks to its architecture that is specifically meant to model temporal sequences without specifying a specific memory length. In the attempt to take into consideration the fact that the nature of the generated sequences is not universal, but it critically depends on the typology of user, we cluster users in groups and learn separate models for each of them. Differently from previous work [32], we cluster users based on their demographics rather than on their past activities, consistently with the intent of obtaining results that are portable to new data without a cold start problem.

The main scientific contributions of this paper are: (1) addressing the problem of next POI category using a machine learning approach on sequences of temporally consecutive check-ins; (2) use of a Recurrent Neural Network (RNN) with Gated Recurrent

[1]The term venue is used interchangeably with POI in this work to describe an entity that has a somewhat fixed and physical extension as defined by http://schema.org/Place

Units (GRU) with multiple layers as a model; (3) an initialization of the vectors fed to the neural networks using an unsupervised feature learning algorithm (node2vec) on the hierarchical graph modelling the Foursquare taxonomy; (4) a user clustering based on demographics that is not affected by the cold start problem; (5) an evaluation protocol based on perplexity, which is new in this domain and is able to address the limitations of using accuracy on a set of interdependent target categories.

2 RELATED WORK

2.1 Venue categories

The availability of venue categories from LBSN data has inspired a number of studies in the past years. In [17], the authors assess the correlations among venue categories and popularity with a statistical study on a large sample of check-ins collected from different geographical regions. In [26], the authors leverage venue categories to automatically create a high level map of the neighborhoods of a city using density-based clustering techniques. In [22], the authors use venue categories to create semantic representation of city neighborhoods and users. In [24], the authors create a semantic representation of a city as a bag of venue categories and use it to define a similarity measure between cities.

2.2 Next POI recommendation

All of these studies, however, do not take into account the temporal dependence among venue categories, i.e. they do not attempt to predict where a user will move next considering the history of her movements in terms of venue categories. This task is similar to the next POI prediction, which has received some attention in the past years. For instance, in [4] the authors propose a matrix factorization method including personalization and geographic constraints that attempts to predict the next check-in of the user based on her past activities and geographical factors. In [8], the authors use a metric embedding approach to develop a personalized model of the sequential transition of POIs. These two studies directly develop a model to recommend the next POI, while, similarly to our approach, in [32] the authors focus on modeling sequences of venue categories. They propose a framework that uses a mixed hidden Markov model to predict the most likely next venue category and recommend POIs belonging to the most likely next category in the neighborhood of the user. Although this work has some common features with the one proposed in this paper, such as the modeling of venue categories transitions rather than directly the POIs transitions, there are important differences. First, they utilize Gowalla's[2] data and venue categorization, which included only nine broad categories, such as *Food* or *Shopping*, which can reasonably considered independent among each other. Our work, on the other hand, is based on Foursquare taxonomy[3], which includes 920 categories organized in a hierarchical fashion, and thus requiring a more complex modeling effort. Secondly, while they cluster users based on their past activities, we follow a different approach, considering user demographics, effectively tackling the new user problem. Third, they use a Hidden Markov Model while we use an approach based on Recurrent Neural Networks. Finally, they evaluate the proposed approach using accuracy, while we use perplexity.

[2]https://en.wikipedia.org/wiki/Gowalla
[3]https://developer.foursquare.com/categorytree

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have received a great deal of attention in machine learning research lately [16], as, thanks to their improved architectures [5, 12] and the advancements in computational power, they are able to effectively model sequences. For this reason, they have been used successfully for tasks such as speech recognition [10], sentiment analysis [30], image captioning [13] and neural language models [20]. One of the typical applications of RNN in the field of language modeling is that of generating text by recursively predicting the next word in a sentence [29]. This task is very similar to the use of RNNs in this work, in which the analogy is that of interpreting a sequence of venue categories coming from users' check-ins as a sentence in a text.

2.4 Itinerary recommendation

The problem of modeling and recommending paths to users share important features with that of itinerary recommendation, which aims at recommending sequences of POIs, considering constraints such as time, budget and personal preferences. Typically, to each POI a score is assigned based on popularity and/or personal preferences, travel times between POIs are inferred from data, and the problem of itinerary recommendation is tackled as an optimization problem where the objective is to maximize the total possible score of the itinerary while complying with the constraints [7, 15, 31]. Note that the greatest difference with our approach is that we do not explicitly formulate the optimization problem with constraints, but rather assume that good paths will be learned from LBSN data.

3 APPROACH

3.1 Problem statement

In this work, we address the problem of next POI category prediction, i.e. we aim to learn to predict the category of the next POI that a user will visit, in order to be able to generate and recommend new paths.

Definition 1. *Given the space of POI categories C, the space of check-in ids I, the space of timestamps T, the space of users U, a check-in is a set $v = \{i, c, \tau, u\}$ where $i \in I$ is the check-in id, $c \in C$ is the category of the POI, $\tau \in T$ is the timestamp at which the check-in has been performed and $u \in U$ is the user who has performed the check-in.*

Definition 2. *A path is an ordered sequence of POI categories $(c_1, c_2, ..., c_t)$, extracted from a sequence of temporally ordered check-ins performed by a particular user $u \in U$, i.e. $(\{i_i, c_i, \tau_i, u\})$ for $i = 1..N$ and where $\tau(i+1) > \tau(i)\ \forall i$.*

Definition 3. *We define a category index $\alpha \in \mathbb{N}$ with $\alpha = 1..|C|$ and that uniquely identifies a category $c^\alpha \in C$.*

In order to learn to generate the next category c_{t+1} of a path, we collect M paths from LBSN and learn a model of the conditional probability $P(c_{t+1}|c_t, c_{t-1}, c_{t-2}, ..., c_1)$ from these sequences of POI categories. Then, from this model, the next category c_{t+1} can be

determined as:

$$c_{t+1} = \arg\max_{c \in C} P(c|c_t, c_{t-1}, c_{t-2}, ..., c_1) \quad (1)$$

3.2 Model

We propose an approach based on Recurrent Neural Networks, which are specifically meant to deal with sequential data. The main difference of RNNs with respect to standard feed-forward neural networks is the presence of a hidden state variable h_t, whose value depends both on the input data presented at time x_t and, by means of loop connections, on the previous hidden state h_{t-1}[9]. A typical application of RNNs in neural language modeling is that of generating text recursively applying a "next word prediction" [29], and in the same spirit we address the problem of next POI category prediction. The main idea is that of using a supervised learning approach where the targets correspond to the inputs shifted in time, i.e. $X = \{(c^j{}_0, c^j{}_1, ..., c^j_{N_j-1})\}$ and $Y = \{(c^j{}_1, c^j{}_2, ..., c^j_{N_j})\}$ where $j = 1...M$ is the path index and N_j is the length of the $j-th$ path. The architecture of the neural network is illustrated in Fig. 1. To simplify the notation, we now drop the path index j and consider one path to illustrate the functioning of the network. A venue category c_t is fed into the network via an encoding into an input vector x_t, which is then passed to a Gated Recurrent Unit. Gated Recurrent Units (GRU) are gating mechanisms that improve the ability of the RNNs to store long sequences and that recently have been proven to be as effective as more complicated architectures such as Long Short-Term Memory (LSTM) units [5]. The update of the GRU unit hidden state, i.e. the computation of the new state h_t given the previous state h_{t-1} and the current input x_t, is described by the following equations:

$$r_t = sigmoid(W_r h_{t-1} + W_r x_t + b_r) \quad (2)$$

$$h'_t = tanh(W_i(r_t \otimes h_{t-1}) + W_i x_t + b_i) \quad (3)$$

$$z_t = sigmoid(W_z h_{t-1} + W_z x_t + b_z) \quad (4)$$

$$h_t = z_t \otimes h' + (1 - z_t) \otimes h_{t-1} \quad (5)$$

where *sigmoid* and *tanh* indicate respectively the sigmoid and hyperbolic tangent activation functions and $\otimes$ represent the element-wise product of the matrices. r is called the 'reset gate' and it allows to forget or remember the previous state h_{t-1} when generating the candidate state h'_t. z is called the 'update gate' and intuitively it controls how much the unit needs to update its state. W_i, W_r, W_z are weight matrices that are learned during the training.

The GRU computes the hidden state h_t which is stored for the next iteration and used to compute the output of the current iteration o_t. Before computing the output o_t, during training time, a Dropout layer is applied. The Dropout layer is a regularization mechanism which, at training time, randomly switches off a fraction p of neurons, called the *dropout rate*, preventing them from co-adapting and overfitting the sampled data [28]. Dropout can be modelled with a mask vector m_t, whose values can be either 1 or 0 with probability p. After the dropout layer, the output state $o_t = tanh(W_o h_t m_t)$ is computed using a fully connected layer whose weights are defined by the matrix W_o, which is learned at training time. W_o is shaped so that the dimension of the output vector is equal to the number of possible categories, i.e. $|o_t| = |C|$. Thus, we can index the components of the output vector using the category index o^α_t. Then, the Softmax layer normalizes the outputs, turning them into a probability distribution over a set of possible outcomes [2]:

$$softmax(o^\alpha_t) = \frac{e^{o^\alpha_t}}{\sum_{k=1}^{|C|} e^{o^k_t}} \quad (6)$$

In this way, the Softmax layer models the probability distribution of the next category:

$$softmax(o^\alpha_t) = P(c_{t+1} = c^\alpha | c_t, c_{t-1}, c_{t-2}, ..., c_1) \quad (7)$$

as o^i_t depends on the current category encoding x_t of c_t, but also on all the previous encodings of the sequence by means of the hidden state h_t. During the training process, we train the network to produce a probability distribution of categories that is as close as possible to that observed in the data, i.e. maximizing the probability of the observed data. Therefore, we define the loss L_t as the cross entropy:

$$L_t = -logP(c_{t+1} = c^\alpha_{t+1} | c_t, c_{t-1}, c_{t-2}, ..., c_1) = -log(softmax(o^\alpha_t)) \quad (8)$$

where c^α_{t+1} is the category observed in the data as $t+1$ element of the path. The loss is optimized using Adam [14], an enhanced version of the stochastic gradient descent that introduces momentum and adaptive learning rates. The gradients of the loss function are computed using back propagation on the unrolled neural network [27]. The model has a number of hyper-parameters, such as the number of neurons in the hidden state n_{hidden}, the number of hidden layers n_{layers}, the learning rate l_r and the number of epochs η. We optimize these hyper parameters using a grid search on a validation set (see Sec. 5).

3.3 Feature learning from category hierarchy

In this work, the space of possible categories C is defined by the Foursquare Taxonomy, which defines and classifies categories in a hierarchical ontology. As can be seen in Fig. 1, it is necessary to specify an encoding to turn the categories into input vectors to be fed into the neural network. A simple and widespread approach to encode categorical variables is that of using the so called *one-hot* encoding, i.e. to use a binary vector whose dimension is equal to the size of the vocabulary $d = |C|$ where only one component is different from 0 using the category index α:

$$x^{one-hot}_k(c^\alpha) = \begin{cases} 1 \iff k = \alpha \\ 0 \iff k \neq \alpha \end{cases}$$

The one-hot encoding is a sparse representation and, although straight forward and intuitive, has a number of shortcoming. First, the size of the input vector depends on the size of the category vocabulary C. This can be defined as the total number of distinct categories that appear in the data, hindering the applicability of the model to unobserved categories, or as the total number of possible categories, which can result in a waste of computational resources when certain categories are not observed in the data. Secondly, the one-hot encoding considers each category as independent from each other and equidistant from the others. Consider as an example the case of three categories: *Restaurant* = (1,0,0), *Italian Restaurant* = (0,1,0), *Movie Theater* = (0,0,1). This representation does not allow to determine whether *Restaurant* is more similar to *Italian Restaurant* or *Movie Theater* and thus fails to effectively represent the hierarchical relations among categories. For this reason, we

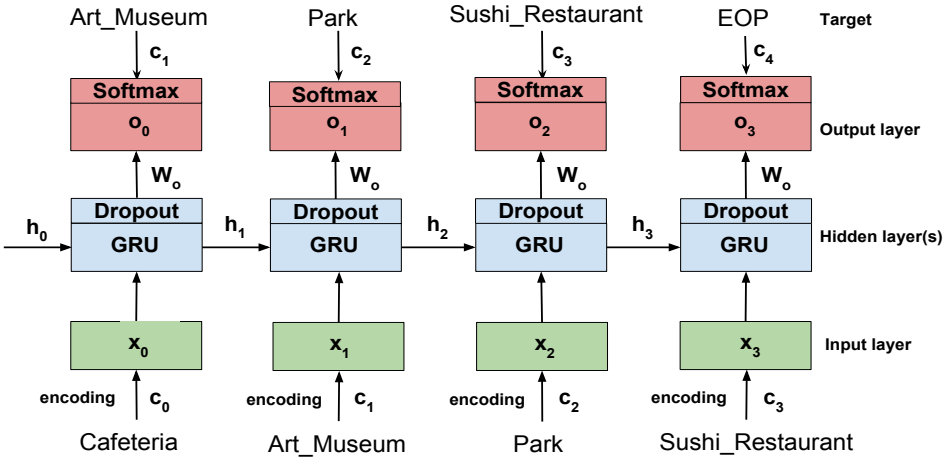


Figure 1: Architecture of the RNN. 'EOP' is a special symbol describing the end of a path.

use node2vec [11], an unsupervised feature learning approach that maps nodes in a graph to a dense vector representation in a Euclidean space of fixed dimension preserving the structure of the graph. Node2vec can be seen as an adaptation of the word2vec model [21] to graphs, as it simulates a random walk on the graph, turning it into an order sequence of nodes, which constitutes the "words" of a document that is then processed using word2vec. In node2vec, we use a uniform exploration, i.e. $p = q = 1$, a dimension of the obtained vector $d = 100$ and 100 walks per category node. In Sec. 5, we compare the results obtained with $x^{one-hot}$ and $x^{node2vec}$, showing that the latter leads to better result and faster computation. A dynamic visualization of the node2vec category embedding can be found online[4]. To obtain a good visualization, we suggest to use TSNE [19] with at least perplexity 20 and 1000 iterations.

3.4 Personalized model

In order to take into account the fact that the next POI category prediction problem can strongly be influenced by personal attributes of a user, we segment the set of users U in a collection of clusters according to the user demographics. Considering the set of languages $l \in L$ and of genders $g \in G$, we generate all possible clusters U_{lg}, U_l, U_g and we segment the whole set of paths P accordingly generating P_{lg}, P_l, P_g. We split each of them into training set and test set containing respectively 80-20% of the data, we train the model on the training sets and assess performance on the test sets.

4 EXPERIMENTAL SETUP

4.1 Data Collection

In order to collect check-ins and sufficient user information to perform a demographic clustering, we use as data sources both the Twitter and the Foursquare API. We collect via the Twitter Search API check-ins done through the Swarmapp[5] application and publicly posted on Twitter, obtaining 1.5M check-ins from 235.6K users in the temporal interval going from 05-04-2017 to 11-04-2017. From this data, we are able to extract for each check-in the language spoken by the user and the id of the check-in. With the check-in id, to gather additional information about the venue and the user, we query the Foursquare API[6], obtaining the venue category c and the user gender. Thus, for each user, we have: $(user\_id, language, gender, (c_1, \tau_1), ..., (c_N, \tau_N))$, where τ is the timestamp of the check-in.

4.2 Preprocessing and Path Extraction

Among all users, only a small part of them uses the application frequently enough to be likely to generate a path. Thus, as a pre-filter to speed up the next processing steps, we filter out users with less than 10 check-ins. We also observe that there are users with a very large number of check-ins, who are likely to be bots. In order to remove them, we develop a heuristic according to which if a user has done more than twice two check-ins in one minute is a bot. By manually checking the results on a sample of 50 users, we observe no false positives. Then, in order to extract the paths, i.e. temporal sequences of correlated venue categories, we apply the principle according to which two check-ins are part of the same path if and only if they both occur within a time window, similarly to what has been done in [7]. Thus, given a set of timestamped check-ins performed by a given user $(c_1, \tau_1), ..., (c_N, \tau_N)$, we split this sequence in multiple paths whenever $\tau_{i+1} - \tau_i > 8h$, i.e. the time difference between two consecutive check-ins is higher than 8 hours. Isolated check-ins, i.e. with $\tau_{i+1} - \tau_i > 8h$ and $\tau_{i-1} - \tau_i > 8h$, are removed from the data. We obtain 29.5K paths, with an average length of 4.2 and a maximum length of 50. In Tab. 1, we report the number of users and check-ins after each preprocessing step.

[4] http://projector.tensorflow.org/?config=https://gist.githubusercontent.com/enricopal/9a2683de61f5fb16c4f59ae295e3fef7/raw/159df72f47e881d0f314096fcc8ea561fb7132b9/projector_config.json

[5] https://www.swarmapp.com

[6] https://developer.foursquare.com/docs/checkins/resolve

	Users	Check-ins
Collection	235.6K	1.5M
More than 10 check-ins	19.5K	400K
Bot removal	12.4K	184K
Path extraction	12K	123K

Table 1: Number of distinct users and check-ins originally and after each preprocessing step.

rank	n_hidden	l_r	epochs	n_layers	ppl
1	64	10^{-4}	5	3	71.333
2	64	10^{-4}	5	2	71.609
3	64	10^{-4}	2	3	71.630
4	128	10^{-4}	2	2	71.645
5	128	10^{-4}	5	2	72.048

Table 2: Perplexity on validation set for the top 5 configuration of hyper parameters.

4.3 User clustering

As we have mentioned in Sec. 1, the way in which tourists explore a city is different and personalized. Although a personalized path recommendation would be desirable, the amount of training data is not sufficient to achieve such a goal. Therefore, we cluster users in groups and tailor the path recommendation to a given user group. In order to obtain results that are general and do not depend on the specific dataset that we have collected, we propose a clustering approach based on the user demographics information that we have collected, i.e. the language and the gender of the user, segmenting the collected paths according to these clusters. We observe 30 distinct languages in the data and count the number of paths per each language-gender pair. We also consider higher level clusters such as: (all, gender) and (language, all), which can be used when only one of the two features about the user is available. We require to have at least 100 users to create a cluster, obtaining 22 distinct clusters.

4.4 Evaluation

In the experimental part of this work, we try to answer to the following research questions:
1) What is the most effective architecture of the RNN model, i.e. what are the best hyper-parameters of the model?
2) Is the dense encoding provided by node2vec more effective than the sparse one-hot encoding?
3) Are Recurrent Neural Networks better at generating paths with respect to a model with a fixed memory window, such as a bigram model?
4) Is the clustering of users favoring or hindering the effectiveness of the model?
In order to answer to these questions, we need to define an appropriate metric to measure the performance of the model. Although accuracy has a straight forward interpretation as it is simply measured as the fraction of correctly predicted venue categories, it would consider all categories independently and weight all errors in the same way. For example, predicting *Sardinian Restaurant* or *Thai Restaurant* when the true category is *Roman Restaurant* would count as an error in the same way. Thus, we opt for a different metric, commonly used in neural language modeling evaluation, that is perplexity [1, 20]. Perplexity is defined as the exponential of the average negative log-likelihood of the model, which in our case becomes:

$$ppl = 2^{-\frac{1}{(\sum_{k=1}^{M} N_k)} \sum_{j=1}^{M} \sum_{t=1}^{N_j} log P(c_{t+1}^j = c_{t+1}^\alpha | c_t^j .. c_1^j)} \quad (9)$$

where M is the total number of paths, N_k is the length of the k-th path and c_{t+1}^α is the category observed in the data as $t+1$ element of the path. Intuitively, perplexity measures the "surprise" of the model in observing the test data. Note that if we roll an ideal die with a number of faces equal to the number of categories C, i.e. $p = \frac{1}{|C|}$, the perplexity is then exactly $ppl = |C|$. Thus, the perplexity can be interpreted as the number of possible outcomes among which a random system should guess. The lower the perplexity, the better is the model. Also note that ppl is equal to the exponential in base 2 of the cross entropy between the model and the data distribution, i.e. the average of the loss L_t over all timesteps of all paths and is thus naturally optimized by the model training. Having defined a metric to evaluate the model, we create a training set and a test set, containing respectively 80-20% of the paths. The validation set used for the optimization of the hyper-parameters is in turn extracted as a 20% of the training set.

5 RESULTS

5.1 Hyper parameters optimization

In order to optimize the hyper parameters (experiment 1), we perform a grid search, i.e. we explore all the possible combinations of the following values:
number of neurons in the hidden layer: $n_{hidden} = [64, 128]$
learning rate: $l_r = [10^{-4}, 5 * 10^{-4}, 10^{-3}]$
number of epochs: $epochs = [1, 2, 5, 10]$
number of hidden layers: $n_{layers} = [2, 3]$
For each configuration $(n_{hidden}, l_r, epochs, n_{layers})$, we train the model and measure its perplexity on validation data, exploring a total of 48 possible configurations. In Tab. 2, we report the best 5 configurations. We can observe that a small learning rate is helping the model learn and that depth, i.e. number of hidden layers, is more effective than width, i.e. number of neurons in the hidden layers. We also observe that training the model for more epochs increases the performance.
In the rest of the section, unless otherwise specified, we use the best configuration of the model.

5.2 Test set scores

We now show the results corresponding to the experiments 2 and 3, i.e. we compare the model initialized with node2vec vectors, with one-hot vectors and the baseline bigram model on the test set. The bigram model is built by estimating the 1-st order transition probabilities $P(c_{t+1}|c_t)$ by counting their normalized co-occurrence

System	ppl
RNN-node2vec	75.271
RNN-onehot	76.472
bigram+smoothing	125.361
random	741

Table 3: Perplexity on test set for the proposed approach and baselines.

frequencies on training data and using add-one smoothing to account for bigrams that do not appear in the training data [3]:

$$P(c_{t+1}|c_t) = \frac{max(1, v_{c_{t+1},c_t})}{|C| + v_{c_t}} \quad (10)$$

where v_{c_{t+1},c_t} denotes the frequency of the bigram (c_{t+1}, c_t), i.e. of co-occurrence of the categories c_{t+1} and c_t whereas v_{c_t} denotes the frequency of the category c_t.

The results are reported in Tab. 3. We can observe that the RNN with node2vec initialization performs better with respect to the other systems and that RNN with one-hot encoding is still far better than the bigram model. This shows, on the one hand, the effectiveness of node2vec as an initialization strategy and that of RNNs in predicting paths. We also observe that the difference between the node2vec and one-hot initialization is small, highlighting the ability of RNNs of learning well also starting from a sparse representation. However, we observe a ratio in computing time of 1.35, as the model runs in 39 minutes with node2vec embeddings and in 53 minutes with one-hot encoding on a server with 48 CPU cores and 256GB of RAM, thanks to the 'compressed' representation of the inputs. In general, we can say that the proposed approach achieves a perplexity of 75.271 on the test set, shrinking of about 10 times the space of possible categories among which a random system would have to guess.

5.3 Personalized model

In this section, we compare the performance of the global model to that of the model trained on the paths belonging to a specific user cluster (experiment 4). The perplexity score, the number of paths, the average path length and the max length are reported for each user cluster in Tab. 4. Note that not all users choose to show their gender or language, as can be verified for example by noting that $Users(M) + Users(F) < Users(All)$. From the results, we can observe that the model perplexity is increasing for certain user clusters and decreasing for others, hinting to the fact that some user clusters might be less predictable than others. For instance, we observe that Turkish speaking users have a much lower perplexity than Dutch speaking users and we might be tempted to conclude that the behavior of the former category is much easier to predict than that of the latter. However, an important role is played by the dimension of the training set, which varies significantly across the clusters and that is a key ingredient in the learning process. In order to look into the correlation between the model perplexity and the number of paths corresponding to the user cluster, we create a scatter plot (see Fig. 2) and measure the Spearmann correlation coefficient among the two variables [6], obtaining a negative correlation $\rho = -0.48$ with a two-sided p-value $p = 0.02$. Both the plot and the correlation coefficient thus appear to show that the amount

Gender	Lang	Paths	avg_l	max_l	ppl
M	All	18,718	4.23	50	75.478
F	All	8,741	4.16	38	73.024
All	En	8,955	3.96	35	71.343
All	Ar	439	4.15	28	100.772
All	Es	1,745	3.79	17	89.994
All	Ja	7,532	5.09	50	84.534
All	Nl	293	4.01	37	112.966
All	Pt	2,713	3.76	23	72.954
All	Th	795	4.19	26	90.343
All	Tr	6,142	3.85	31	66.838
F	En	2,636	3.94	28	74.999
F	Es	480	3.81	17	71.223
F	Ja	2,290	4.89	37	98.801
F	Pt	780	3.79	22	78.355
F	Th	234	4.05	15	83.601
F	Tr	1,863	3.80	18	82.078
M	En	5,771	3.95	35	69.481
M	Es	1,164	3.74	17	84.008
M	Ja	4,726	5.18	50	90.437
M	Pt	1,778	3.76	23	70.189
M	Th	524	4.30	26	94.132
M	Tr	3,886	3.89	31	69.056
All	All	29,465	4.20	50	75.271

Table 4: Personalized model vs global model.

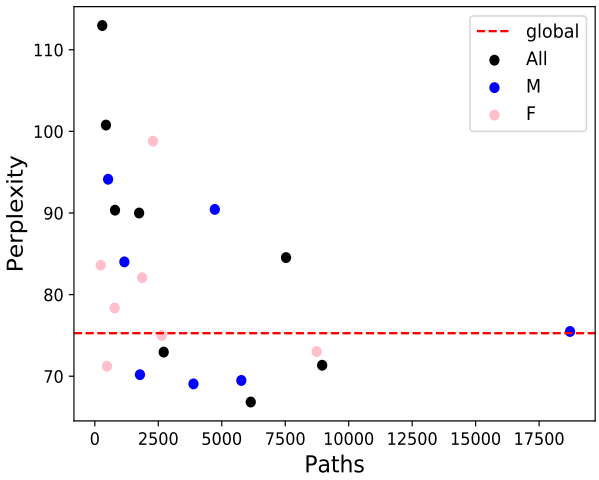


Figure 2: Number of paths in user cluster U_i and model perplexity. Blue circle correspond to "M", pink circles to "F" and black circles to "All". The horizontal dashed line correspond to the global model, including all users.

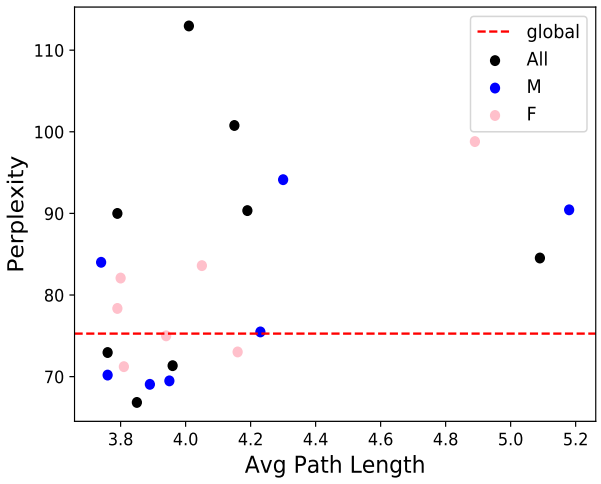


Figure 3: Average path length for user cluster U_i and model perplexity. Blue circle correspond to "M", pink circles to "F" and black circles to "All". The horizontal dashed line correspond to the global model, including all users.

of training data is negatively correlated with the perplexity of the model, supporting the intuition that the size of the training set is actually enhancing the model.

Another factor that might influence the performance in terms of next category prediction across the different user clusters is the fact that the average path length varies. Similarly to the previous analysis, we plot average path length and model perplexity (see Fig. 3) and measure the Spearmann correlation between the variables, obtaining a position correlation $\rho = 0.49$ with a two-sided p-value $p = 0.02$.

6 CONCLUSIONS

In this paper, we propose a novel approach to recommend sequences of POI categories, as a first step to create a system that is able to automatically learn from data a personalized tourist path. The approach is based on a Recurrent Neural Network model, which shows to be able to model and predict effectively sequences of POI categories. We experiment different hyper parameters of the architecture of the network, showing the importance of a small learning rate and of stacking up multiple layers rather than increasing the number of neurons in the hidden layers. We also show that initializing the categories using an encoding based on node2vec improves the performance of the model with respect to the standard one-hot encoding, both in terms of model perplexity and of computing time. The analysis of the results of the model using different user clusters has a less definite interpretation, as we observe that in certain cases the performance increases and in other cases the performance decreases. We suggest that possible biasing factors are the size of the training set and the average path length, which is confirmed by a correlation analysis with the perplexity of the model. Further studies will extend the analyses to a larger dataset with a larger sample of user to rule out finite size effects on the performance of the clustered models and to include a larger temporal interval to exclude from the analysis possible seasonal effects.

Future work will also involve the integration of the next POI prediction into a real recommender of sequences of POIs for tourists. Given the next most likely POI category, a short list of sorted POIs belonging to that category will be retrieved from a knowledge base containing places and events. The short list will be computed using entity2rec [23] leveraging user context (e.g. geographical position), inherent venue peculiarities (e.g. ratings, reviews) and user preferences.

ACKNOWLEDGMENTS

This work was partially supported by the innovation activity PasTime (17164) of EIT Digital.

REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.
[2] Christopher M Bishop. 2006. Pattern recognition. *Machine Learning* 128 (2006), 1–58.
[3] Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 310–318.
[4] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation.. In *IJCAI*, Vol. 13. 2605–2611.
[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[6] Gregory W Corder and Dale I Foreman. 2014. *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons.
[7] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. 2010. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*. ACM, 35–44.
[8] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation.. In *IJCAI*. 2069–2075.
[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.
[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[13] Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3128–3137.
[14] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] S Kotiloglu, T Lappas, K Pelechrinis, and PP Repoussis. 2017. Personalized multi-period tour recommendations. *Tourism Management* 62 (2017), 76–88.
[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[17] Yanhua Li, Moritz Steiner, Limin Wang, Zhi-Li Zhang, and Jie Bao. 2013. Exploring venue popularity in foursquare. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 3357–3362.
[18] Bin Liu and Hui Xiong. 2013. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 396–404.
[19] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[20] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.
[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In

Advances in neural information processing systems. 3111–3119.

[22] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. 2011. Exploiting Semantic Annotations for Clustering Geographic Areas and Users in Location-based Social Networks. *The Social Mobile Web* 11, 2 (2011).

[23] Enrico Palumbo, Giuseppe Rizzo, and Raphael Tröncy. 2017. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *Proceedings of the 11th ACM conference on Recommender systems*. ACM.

[24] Daniel Preoţiuc-Pietro, Justin Cranshaw, and Tae Yano. 2013. Exploring venue-based city-to-city similarity measures. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*. ACM, 16.

[25] Giuseppe Rizzo, Rosa Meo, Ruggero G Pensa, Giacomo Falcone, and Raphaël Troncy. 2017. Shaping City Neighborhoods Leveraging Crowd Sensors. *Information Systems* 64 (2017), 368–378.

[26] G. Rizzo, R. Troncy, O. Corcho, A. Jameson, J. Plu, J.C. Ballesteros Hermida, A. Assaf, C. Barbu, A. Spirescu, K. Kuhn, I. Celino, R. Agarwal, C.K. Nguyen, A. Pathak, C. Scanu, M. Valla, T. Haaker, E.S. Verga, M. Rossi, and J.L. Redondo Garcia. 2015. 3cixty@Expo Milano 2015: Enabling Visitors to Explore a Smart City. In *14^{th} International Semantic Web Conference (ISWC), Semantic Web Challenge*.

[27] Jrgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[29] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 1017–1024.

[30] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification.. In *EMNLP*. 1422–1432.

[31] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10.

[32] Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. What's your next move: User activity prediction in location-based social networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 171–179.

[33] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 312–321.

A Delay-Robust Touristic Plan Recommendation Using Real-World Public Transportation Information

Victor A. Arrascue Ayala
*Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110
Freiburg, DE
arrascue@cs.uni-freiburg.de

Kemal Cagin Gülsen
*Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110
Freiburg, DE
guelsenk@cs.uni-freiburg.de

Marco Muñiz
+Department of Computer Science
Aalborg University
Fredrik Bajers Vej 5, DK-9220 Aalborg
East, DK
muniz@cs.aau.dk

Anas Alzogbi
*Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110
Freiburg, DE
alzoghba@cs.uni-freiburg.de

Michael Färber
*Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110
Freiburg, DE
michael.faerber@cs.uni-freiburg.de

Georg Lausen
*Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110
Freiburg, DE
lausen@cs.uni-freiburg.de

ABSTRACT

Tourism Recommender Systems (TRS) assist tourists in designing a plan for a soon-to-be visited city, which consists of a selection of relevant points-of-interest (POI), the order in which they will be visited, the start and end time of the visits, etc. These tools filter POIs based on the tourist's preferences and take into account time constraints, like the desired duration of the plan, or the POI's opening or closing times. However, being able to provide tourists with an additional travel plan which explains how to reach those POIs using public transportation is a feature in which TRSs come short. Existing solutions try to solve the problem in a simplified way and do not model all possible events involved in using public transportation, such as combining transfer times and trips, changing vehicles, or dealing with delays of transportation units. We therefore propose three novel approaches to generate visit plans and their corresponding travel plans, namely SILS, TRILS and PHILS, which overcome these weaknesses. These approaches generate visit plans by iteratively adjusting them according to the traveling information and differ in the way the adjustment is done. Our experiments on a real-world POI dataset and public transportation information of the city of Izmir show that our approaches outperform the state-of-the-art in terms of quality of recommendations. Moreover, they are also able to provide both visit and travel plans in real-time and are robust in case of delays. To the best of our knowledge previous approaches have not been able to achieve this level of practicality.

KEYWORDS

Tourism Recommender Systems, Tourist Trip Design Problem, Time-dependent Orienteering Problem with Time Windows, Iterated Local Search, Route Planner, Delays.

1 INTRODUCTION

Visiting touristic attractions, walking through historical places, or trying local food are among the main activities tourists undertake when they visit a new city or country. Given that the number of attractions is typically large and that tourists are also restricted to time and/or money budgets, they need to optimize and sometimes compromise on the selection of relevant attractions. In addition to this, they also have to figure out if the place is reachable and eventually find a feasible way to reach the location using public transportation. This process can be very cumbersome. Therefore, Tourism Recommender Systems (TRS) have been developed for assisting tourists in planning their trips.

In the literature the problem of generating a visit plan (no public transportation involved) is known as the *tourist trip design problem (TTDP)* [10], while the attractions are referred to as points-of-interest (POIs). This problem is formally defined as follows: given a set of POIs $p_1, ..., p_n$, each POI is associated with some *profit* s_i, which reflects the user's affinity towards this POI. The goal is to find $V = (p_i, p_j, ..., p_l, p_k)$, a sequence of POIs to be visited in a given order (i.e. a *visit plan*) which maximizes the collected profits taking into account the following time constraints: (1) the user's specified time budget, (2) the availability of POIs (e.g. opening hours), and (3) the travel time required to move from one POI to the next one.

In this paper we aim at additionally integrating public transportation information to assist the user in traveling from one attraction to the next one. This feature is indeed perceived by tourists as one of the most useful functionalities [6]. However, the biggest challenge when tackling the TTDP problem including also public transportation information is the increased level of difficulty of constraint (3), i.e. the fact that travel times can significantly vary depending on the situation, e.g. the timetables, the time required to reach the closest station by foot, etc. This effect is known as *time-dependency*. Formally, in addition to the visit plan V, we would like to generate $T = (t_{(i,j)}, ..., t_{(l,k)})$, the sequence of travel plans, each indicating how to move from one attraction to the next one, e.g. from p_i to p_j, according to V using public transportation. Examples of those instructions could be how to reach the departure station from an attraction by foot, at what time to leave the attraction, in which station to get out from a bus/tram, when to change service type, etc. Moreover, both plans V and T should be provided in real-time (time response below 5 secs).

Surprisingly, integrating public transportation information is still quite unexplored [10]. This is likely due to the associated computational challenges and the need for real-time data. Note that a TRS of this kind needs to consider (i) the public transportation system's status itself (e.g. buses and trams may have delays) as well as (ii) the point in time when the user departs to the next attraction[1] (e.g. if a bus cannot be reached any more, the user needs to wait, which stretches the travel time and which might lead to the fact that a planned attraction cannot be visited any more). The TTDP problem with public transportation becomes even more complicated when considering changing buses etc.

Existing solutions for TTDP, which integrate public transportation information (and therefore, with the time-dependency constraint) such as [6, 8, 11, 25] try to solve the problem in a simplified and therefore not realistic way. They can be grouped as follows: (1) Approaches that get rid of the time-dependency by considering a time-independent approximation of the problem, e.g. by using average travel times. However, trip plans computed with average travel times differ in practice significantly with respect to solutions with real travel times [25]. (2) Approaches that pre-compute all travel times for all possible pairs of POIs and times. However, pre-computing all possible travel times, considering all possible combinations of POIs and times is clearly feasible only for small transportation networks or small cities [6]. (3) Approaches that pre-compute travel times exploiting regularities in the schedules. However, the assumption of periodic service schedules does not hold in realistic urban transportation networks [11]. (4) Approaches that sacrifice some route planning aspects, e.g. the multi-modal feature which allows us to model different types of transportation services, or transfers which allows us to model changes between transportation services in a route.

It is important to note that incorporating real-time transportation information of a city and generating a travel plan T with detailed instructions to move along POIs, even in case of delays, is not provided by any approach so far, to the best of our knowledge. We therefore propose such an approach which generates a sequence of attractions to visit, together with a travel plan with concrete instructions for the user. In total, we make the following contributions:

(1) We combine i) an approach for generating a sequence of attractions to be visited, using the time constraints regarding attractions and total trip time (i.e. solving the TTDP problem and using the time-dependency constraint for public transportation) with ii) an approach for generating a realistic, delay-aware travel plan. We employ three different strategies (sections 5.1 to 5.3). The travel plan generation is designed to be realistic, as it uses the real-time transportation system's information (departure times with potential delays) and the current place and time of the user. Furthermore, the real-time computation constraint is considered.

(2) We evaluate our three approaches extensively with a large real-world data set, namely the POIs and the public transportation information of Izmir, Turkey. This data set contains 75 POIs, and a large transportation network which consists of approx. 8K stations and 26K bus runs per day. Our evaluation results show that the designed strategies outperform the state-of-the-art in terms of quality of recommendations.

Our approach is applicable to tourist plan recommendation in any city or location in which data about the attractions, their locations, and availability times are available, as its real-time public transportation system information.

The rest of the paper is organized as follows: First we give an overview of the TTDP and the approaches to solve that problem in Related Work (section 2). Then we present the Task Description (section 3) and the models our approaches are based on. We explain the basic concepts of the state-of-the-art method to solve the TTDP, the Iterated Local Search (ILS), in section 4. Section 5 describes our main contribution, three approaches built on top of ILS, i.e. SILS, TRILS, and PHILS, to produce feasible and realistic trip and travel plans. The experiments are shown in section 6. Finally, conclusions and future work are presented in section 7.

2 RELATED WORK

The tourist trip design problem (TTDP) is the problem of generating a sequence of the most relevant POIs to visit without violating user restrictions such as time budget. Although the TTDP has been widely investigated, a gap remains between theoretically solving TTDPs and applying them in practice [26]. One example is the incorporation of public transportation information into the trip plan. In fact, both creating travel plans for POIs and generating travel plans incorporating public transportation are separately considered to be hard to solve and have their own challenges in terms of modeling and computation. In addition to this, the solutions have to be computed in real-time.

One of the earlier works which addresses the TTDP problem is the one by Tumas and Ricci [20], which provides a set of ranked routes (using public transportation) fixing a start and end location. The route itself might lead through famous POIs. While this involves the user in selecting her preferences, the route selection remains the central concept.

However, most of the solutions seen in the literature consist of modeling the problem as extensions of the *Orienteering problem (OP)*. The OP is a combination of two classical problems: the Knapsack Problem and the Traveling Salesman problem. This problem can be defined formally as a graph in which nodes represent POIs and the edges a feasible travel route from one POI to another one. Each node has a "profit 'score" (henceforth simply "profit") that the user can collect by visiting the POI and the edges are weighted with travel times. The OP and its variations cannot be solved in polynomial time and therefore most of the solutions proposed are based on meta-heuristic algorithms which provide near-optimal solutions [10]. Among the existing OP extensions, the best modeler of the usage of public transportation is the *Time-dependent Team Orienteering Problem with Time Windows (TDTOPTW)* [14, 23]. The term *time-dependency* (TD) reflects the fact that the travel time to

[1] In this paper, we assume that the user does not want to leave the attractions earlier than planed, as this might stress her and as it would make the problem hardly manageable.

move from one POI to the other depends on the departure time, e.g. on the schedule of buses, trams, etc. The "*Team*" (T) extension allows one to model trip plans for multiple days. Moreover, the *Time Windows* (TW) represent the fact that visits are limited by opening and closing times of the POI. Algorithms based on Iterated Local Search (ILS) heuristics are considered the state-of-the-art to solve OP and their extensions [14]. An ILS approach for solving extensions of OP typically consists of two operations: INSERT, which inserts a POI into the solution, and SHAKE, which removes a POI to escape from local optima. The most popular version of ILS is probably the one proposed by Vansteenwegen [21].

State-of-the-art for TD(T)OPTW. Garcia et al. [6] were the first ones who tried to address the TDTOPTW problem using real public transportation data from the city of San Sebastian. Their ILS metaheuristic is built upon that of Vansteenwegen [21]. To deal with time-dependency they implement two approaches. The first one consists of using average travel times in their ILS. Since average travel times are not always accurate, they propose an approach to adjust the plan according to real travel times in case this is infeasible, i.e. if the time windows of the nodes included in the solution are violated [11]. This might lead to the removal of some of the attractions, reducing then the overall profit. The second approach is based on a fast local evaluation of the possible insertions. They design three variants: 1) direct public transportation without transfers; 2) an approach based on a pre-calculation which considers transfers; 3) an approach in which transfers are modeled as direct connections. The first two variants fulfill the real-time response requirement. However, they do not realistically model public transportation, because in large networks like cities, it is not always possible to reach a place without transfers, nor are the schedules always regular. To deal with the time-dependency Gavalas et al. extend in [11] a previous cluster-based meta-heuristic approach called CSCRoutes to deal with TDTOPTW. This results into two new approaches, TDCSCRoutes and SlackCSCRoutes. These do not make any assumptions about periodic schedules. In a subsequent work [8] they integrate multimodality into the routing logic as well as other features, such as the possibility of incorporating lunch breaks and the support of arbitrary start and end itinerary locations. Travel instructions are also included. The approach is validated with metropolitan transit network information and real POIs from Athens and Berlin. Verbeeck et al. [25] extend a previous approach [24] to solve the TDTOPTW problem. They use an ant colony system (ACS) based algorithm which constructs several independent solutions. The road network data used consists of data sent by taxis, commercial vehicles, and private cars, rather than public transportation so it is very unlikely that they model transfers. Moreover, they consider some kind of regularities by dividing a day into k slots. Then the set of time-dependent travel times is calculated by repeatedly using a modified version of Dijkstra's algorithm with a departure time equal to the start of a time slot.

Further extensions. The TDTOPTW is not the only extension of TOPTW which tries to model more realistic scenarios. For example POIs are typically treated as points. However, in pratice these might represent large areas such as market areas or neighborhoods in which tourists might require a walking route. Therefore, Gavalas et al. [7] extend the TOPTW problem to incorporate scenic walking routes for exploring tourist destinations and call the new model MTOPTW. Vansteenwegen et al. [22] extend TOPTW by adding constraints that allow POIs to have multiple time windows, e.g. windows which differ on different days. The first extension which aims at generating plans for groups of tourists was proposed by Sylejmani et al. [19]. They extend the Multi Constraint TOPTW (MCTOPTW) to MC-Multiple-TOPTW (MCMTOPTW) to model the multiple trips and tours for tourist groups. Other approaches model congestions or other events which might affect the travel times between nodes. Sometimes these events are difficult or even impossible to predict in a deterministic way [14]. These models are therefore called Stochastic OPTW (SOPTW) and are related to vehicle routing models [15].

Other approaches to solve TTDP. While the most popular version of ILS is the one proposed by Vansteenwegen [21], other variants have been proposed [12, 17]. In [12] they extend the ILS with further operations, namely SWAP, INSERT, ACCEPTANCE CRITERION, and TIMELIMIT.

Completely different strategies have been suggested as well, in addition to these ILS heuristics. In [9] another near-optimal heuristic approach called DailyTRIP was proposed. Other examples include a Tabu Search meta-heuristic [18], fireworks algorithm [5], and simulated annealing [13, 16]. Bitonto et al. [4] model the problem of building itineraries for tourists addressing a Constraint Satisfaction Problem (CSP) by means of the transitive closure. The work of Wörndl [26] is particularly interesting, since they try to solve the whole problem with a modified version of Dijkstra's algorithm, which not only finds shortest paths but also solves TTDP. To do so they maximize the quotient of entertainment divided by distance for each subpath (entertainment is the sum of the scores of all venues along the path). An extended version called constraint-based takes into account the time and budget constraints for the route. None of these approaches directly deals with public transportation.

3 TASK DESCRIPTION

3.1 Overall Task (TTDP-TI)

Figure. 1 illustrates the desired result, i.e. plans, V and T. In order to explain the underlying models used in our approach we need to provide some fundamental definitions. The POIs part of the visit plan V are selected among the set of available POIs $P = \{p_1, p_2, ..., p_n\}$. Each POI has a time window $[o_i, c_i]$ with *opening time* o_i and *closing time* c_i. Therefore, visits should take place within the time window. There is a variable for each POI p_i which models if the user visits it or not. Let v_i be this variable: $v_i = 1$ if p_i is visited and included in the plan V, 0 otherwise. Let t_i^v be the time the tourist u spends at p_i, if $v_i = 1$. If the visit takes place, this time is fixed, i.e. each POI has a recommended visit time and we assume for simplicity this cannot be shortened or extended. Let t_i^s be the start time of a visit at p_i. Then, the ending time of a visit is simply given by $t_i^e = t_i^s + t_i^v$. The time required to travel from p_i to p_j is $w_{(i,j)}^t$. This times depends on the departure time t from p_i. When a tourist visits a POI p_i she collects the *profit* s_i. Furthermore, let $x_{(i,j)}$ be a variable which models the fact that u visits p_j after she visited p_i. If this occurs, then $x_{(i,j)} = 1$, and 0 otherwise. The overall visit time for a plan should not exceed the tourist's total available time. Let t_{max} be the time budget of u. The goal is to generate a visit

plan V together with a travel plan T (which includes the detailed instructions tailored for public transportation). The time-dependency itself is modeled within $w^t_{(i,j)}$. Therefore, we call this problem the *tourist trip design problem with travel instructions (TTDP-TI)*. This can be divided into two subtasks: (1) the generation of a visit plan V taking into account the time-dependency, and (2) providing the travel instructions. Each of these subtasks will be explained in the following subsections.

3.2 Part 1: TTDP

The TTDP part of our problem is modeled as the time-dependent orienteering problem with time windows (TDOPTW). The goal is to produce a visit plan V, i.e. a route r which goes through some of the available POIs, thereby collecting as much profit as possible (*objective function*). For this problem one POI is required as the starting point and one as the ending point of the trip. Typically p_1 and p_n are picked as start and end locations [14, 23]. The problem is subject to a given set of constraints:

(1) A tour starts and ends at two fixed POIs $\in P$.
(2) A tour is a path which connects all visited POIs $\in V$. Each POI included in V should be visited at most once.
(3) The waiting time before a visit at a POI starts is limited by a constant M.
$t^e_i + w^t_{(i,j)} - t^s_j \leq M(1 - x_{(i,j)}), \quad (i, j = 1, ..., n).$
(4) The total time spent for the trip which includes the visit times, travel times and eventually the waiting times before a visit takes place should be less than the specified time budget t_{max}.
(5) The visits take place within the POI's time window.

In the same way as OP, TDOPTW can be formulated in two possible ways: as a graph or as integer linear programming problem. We refer to Vansteenwegen et al. [21] for a linear programming formulation of this problem.

The sequence $V = (p_i, ..., p_k)$ is reconstructed from the variables $x_{(l,m)}$, each of which represents a consecutive visit. In addition to V the visitation times for the each POI, $((t^s_i, t^e_i), ..., (t^s_k, t^e_k))$ are returned. It is important here to notice that getting the travel plans T (i.e. how to reach POIs) are beyond the scope of TDOPTW. They are generated by a separate algorithm, if needed. Most of the solutions relax the travel time $w^t_{(i,j)}$ by removing the time-dependency: $w_{(i,j)}$. Therefore, when the plan is adjusted according to real travel times, this might become infeasible (see section 5).

One important aspect of the TTDP is to determine the importance of a POI for a user u, i.e. to estimate the profit s_i the user could get by visiting the POI. This can be done in different ways.

3.3 Part 2: Route planning for public transportation

In order to obtain the travel instructions T we make use of a route planner in real-time. In fact, designing approaches which provide travel plans has been one of the main concerns of route planning. Not only the time-dependency, but also the real-time requirement are aspects which have been widely studied in this field. To fulfill the real-time requirement graphs are the most recurrent models in public transportation [2].

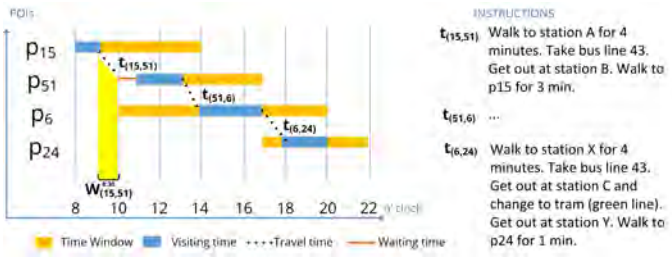


Figure 1: Example for solving the TTDP-IT problem

Our route planner has two key features: the time-dependent model (TDM) together with a state-of-the-art speed-up technique called *transfer patterns* [1]. The route planner is able to provide the path of minimum cost between two locations in the order of milliseconds. Note that without a route planner with these characteristics it would not have been possible to couple the TTDP approach with the travel information. This model supports walking (between stations or to change vehicle), transfers, waiting times at stations, etc. Delays of transportation units are also supported, although the optimality of the solution is no longer guaranteed. However, empirical studies show that transfer patterns are a very robust technique in the presence of delays and leads to suboptimal solutions in less than 3% of the cases in large networks [3].

4 ITERATED LOCAL SEARCH

We now come back to the TTDP problem, which is modeled in our case as a TDOPTW. In this regard the Iterated Local Search (ILS) heuristic is the state-of-the-art to solve TDOPTW. Our three approaches, SILS, TRILS, and PHILS are built upon ILS as conceived by Vansteenwegen [21] and García [6]. Our approaches will be presented in section 5 with a special focus on the novelty aspect. In this section, we will explain the basic concepts underlying the ILS algorithm, which are necessary to explain our approaches. The goal of an ILS algorithm is to return a near-optimal visit plan $V = (p_i, ..., p_k)$. Solutions are built by applying iteratively two operations: An *insertion step*, which inserts a POI into the solution, and a *shake step*, which removes a POI from it. The purpose of the latter is to escape from local optima. The stop condition of ILS requires that a solution is not improved for a certain number of iterations. The time-dependency plays a key role when inserting a new POI into the solution, because the exact time in which the visit starts depends on the departure from the previous POI. In the following, we explain these two steps and the ILS algorithm that combines them.

Insertion Step. In the insertion step, a new visit (POI) is added to the tour. The POI with the highest $Ratio_i$ is always the one picked for the insertion and we calculate $Ratio_i$ as follows:

$$Ratio_i = (s_i)^2 / Shift_i \tag{1}$$

where s_i is the profit, and $Shift_i$ is the extra time added to the duration of the trip plan if p_i is added to it. To compute the extra time, not only the visiting time of p_i is taken into account but also the travel time from the previous POI and to the following POI in the sequence. Since time is limited by the time budget t_{max}, every

time we insert a new POI it is checked whether existing visits still fit in their corresponding POI's time window.

Shake Step. The shake step removes at least one visit from the given tour. The purpose of this step is to escape local optima. The soon-to-be-removed POI is selected in a random fashion by means of variables which rotate over the whole visit plan. The visits scheduled after the removed POI are shifted towards the beginning to avoid unnecessary waiting times. Some of the visits may not be shifted because of the time window constraint. The visits after those non-shiftable visits remain unchanged.

For further details about the used variables and how these are updated in both the INSERT and SHAKE steps, we refer the reader to [21].

ILS: combining both steps. A pseudo-code showing the key steps of the ILS can be found in [21]. The search for a solution is performed until no better solution is found for 150 iterations[2]. At the beginning visits are inserted one by one using $Ratio_i$ until it is not possible to add more of them. This plan is then stored as the current best solution, using the variable *BestSolution*. Another variable *NoImprovementCounter* keeps track of the number of iterations in which no improvement could be achieved with respect to the current best solution. In the next iterations another produced visit plan might be compared with the last best solution found and if this is beaten, *BestSolution* is updated and *NoImprovementCounter* is reset. After checking the new solution, the shake step is applied.

The insertion operation deals directly with the time-dependency, because the travel time from the previous POI has to be considered in order to place the inserted POI at the right position in time. This might lead to the false belief that this information can be directly requested from the route planner every time an insertion is carried out. However, in practice too many visits are inserted and therefore the number of requests sent to the route planner would be too high to keep the whole computation of ILS operating in real-time.

Therefore, works like that of García et al. [6] either use pre-computed travel times, e.g. average travel times, or leave some events, such as transfers (changing buses in a trip), aside. We adopt instead a different strategy, which is outlined in section 5.

5 OUR APPROACHES

A method to produce a visit plan V based on pre-computed values might differ from the real travel times provided by a route planner. Therefore, a visit plan might have to be adjusted, i.e. the travel time between POIs has to be corrected. This might cause an increase in the waiting time or a shift of the visit times. If the time window of any of the POIs included in the solution is violated, then the plan is said to be *infeasible* [11] and it requires a repair strategy. García et al. [6] propose a simple repair strategy: if the real travel time is larger than the average one, some visits have to start later. If this causes a visit to become infeasible the POI is removed from the route. This means that in the best case the profit remains the same. Otherwise some profit is lost.

We therefore designed three strategies which not only provide feasible visit plans but also avoid sacrificing profit. The novelty aspect lies in the fact that we dynamically adjust the visit plan according to real travel times obtained from the route planner. Our heuristics are similar to those of Vansteenwegen [21] and García [6] in the sense that an optimal solution is first computed using average travel times for efficiency. However, while searching for an optimal solution the adjustment takes place by shifting some visits forwards or backwards in time, repairing the plan if required, and then letting the ILS continue with the search. This process allows our ILS-based heuristics to potentially find a different near-optimal solution, while keeping the plan realistic.

[2]This value has been empirically found in the experiments of Vansteenwegen and is a good trade-off between the execution time and the quality of the solution. In that setting the results of the ILS metaheuristic deviate from the optimal solution by only 1.8% using only 1 second of computation [17].

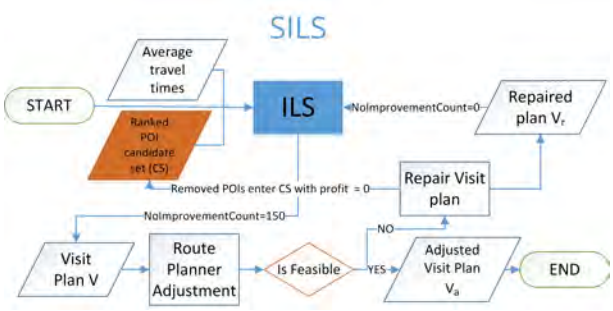


Figure 2: SILS Logic

5.1 Strict ILS (SILS)

The logic of SILS is illustrated in Figure 2. The intuition behind this heuristic is as follows: When *NoImprovementCounter=150* the found visit plan is adjusted with respect to the real travel times returned by the route planner. If after this adjustment the visit plan is feasible, this is returned as the solution (V_a).

Otherwise, the plan is repaired by removing the POIs in which the time window constraints are violated. The repaired solution V_r is then given to the ILS heuristic, which continues the search for an optimal solution from the plan. In addition, the removed POI(s) are penalized and relocated among other POI candidates with a profit $s_i = 0$. This basically disables the POI, which cannot be selected anymore ($Ratio_i = 0$). Moreover, the *NoImprovementCounter* is set to 0.

The process is repeated until a feasible visit plan is returned. Note that this approach can fail if all POIs are disabled. However, this never occurred in our experiments.

5.2 Time-relaxed ILS (TRILS)

The logic of TRILS is illustrated in Figure 3. Excluding one or more POIs from the ILS computation might not help in situations where the average travel time is a bad estimator of the actual travel time. This is the case when the timetable is not very regular or when the variance of the travel times in a day is too large.

Therefore, this approach rather gradually increases the estimated average travel times every time a solution is infeasible by multiplying it with a constant ($step = 0.05$). In this way the estimated travel times can be greater than the real travel times at some point and therefore a feasible plan can be returned. The downside of this strategy is that it might exclude more compact solutions with a higher profit.

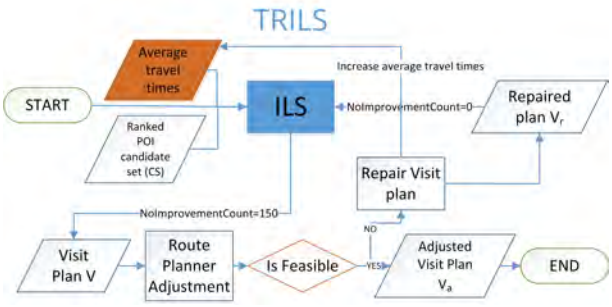


Figure 3: TRILS Logic represented as flowchart

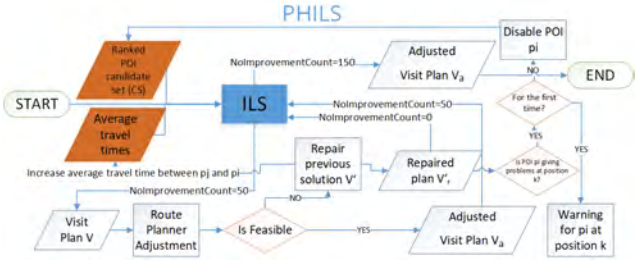


Figure 4: PHILS Logic

As in SILS, TRILS checks the validity of the solution when *NoImprovementCounter=150*. The visit plan is validated against the real time travels returned by the route planner. If after the adjustment, the visit plan is feasible, this is returned as the solution (V_a). Otherwise, the plan is repaired by removing the POIs in which the visits violate the time window constraints. In addition to this, the average travel times are increased by a constant. The first time an infeasible solution is returned, the average travel times are multiplied by 1.05. The second time by 1.10, and so on. The ILS heuristic continues the computation using the repaired plan V_r. It is important to notice that although the profits of the removed POIs remain unchanged, the insert operation might exclude them because of the increased average travel time, which potentially reduces the number of time slots into which these can be potentially inserted.

The process of increasing the average travel times is repeated until eventually a feasible solution is found. The approach fails if the average travel times are increased 5 times. However, this never occurred in our experiments.

5.3 Precise Hybrid ILS (PHILS)

The logic of PHILS is shown in Figure 4. This approach combines the ideas of the two previous approaches in a more fine-grained fashion. First, this approach tries to validate the current best solution when *NoImprovementCounter=50*, i.e. at an earlier stage of the solution computation. Let V be the best known plan at this point of calculation. If the alignment with the route planner does not cause the visit plan to be infeasible, then the ILS continues the search for a better solution. If no better solution is found until *NoImprovementCounter=150* then the visit plan V_a, which is already adjusted, is returned as the solution.

In contrast, if the adjustment causes the plan V to be infeasible three measures are taken: (1) if p_i is the POI in $V = \{..., p_j, p_i, ...\}$ in which the time window constraint is violated, then the average travel time between p_j and p_i is increased. Note that this correction is done for only a single pair of POIs and not for all possible pairs as in TRILS. (2) Instead of letting the ILS continue the search from the repaired plan V_r, the previous best known solution is retrieved. Let V' be this solution. (3) If V' contains p_i, then this is removed from it and this solution is used as the new starting point. The intuition is that it is better to correct the solution which lead to the infeasible plan, rather than the plan itself. Then, the ID of p_i and the position k in the sequence at which it failed are stored. If p_i disrupts another solution at the same position, p_i is permanently removed from the candidate set. Note that p_i is disabled when the failure is produced at the same position k in the sequence and not simply when it occurs, as in SILS. The ILS continues the search for a better solution based on the repaired plan V'_r. The process is then repeated until a feasible visit plan is produced.

5.4 Travel instructions

To generate the travel instructions T required to move from one POI to the next one in the sequence, the route planner can simply store the travel plans (including the travel instructions) of the last adjusted visit plan V. In fact, the last-adjusted plan is also the final returned solution in all three strategies.

To summarize. The three approaches are able to produce feasible solutions, in contrast to approaches based on ILS and average travel times. On the one hand, in the case of SILS and TRILS, the solution is adjusted according to the real travel times before it is returned as the final solution. If the adjusted plan is infeasible, some corrections are being made in the selection strategy, i.e. disabling an out-of-window POI, or increasing the average travel times, respectively. A new optimal solution is then searched iteratively on top of that repaired plan. On the other hand, PHILS combines both strategies in a more fine-grained manner. The travel instructions are provided by the route planner without the need for further calculations.

6 EXPERIMENTS

The ultimate goal of Recommender Systems is to maximize the user satisfaction. Solutions which model the TTDP as TDOPTW or similar extensions assume that profit is a good measure of the user's satisfaction. Therefore, they are compared based on their collected profit under the same constraints. We follow the same line of evaluation. An additional aspect we assess is whether our approaches are able to provide a plan in real-time.

6.1 Set up

Our POI dataset for Izmir consists of 75 POIs in total. POI information like the labels used to build both the user and POI profiles are obtained from Foursquare[3]. The POIs are distributed in 4 Regions (see figure 5): 36 are located within the city center; 17 POIs in the north; 8 POIs east; 11 POIs south-west and 3 POIs outside of the city

[3]https://developer.foursquare.com/

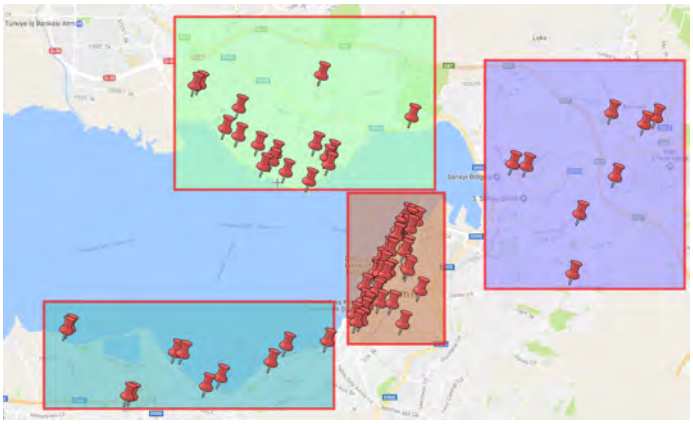


Figure 5: Distribution of POIs in Izmir

(not shown in the map). The General Directorate of ESHOT[4], the public bus transportation corporation of the Municipality of Izmir, Turkey kindly provided us with the public transportation data upon request. All POIs can be reached using the public transportation network. The network consists of 7788 stations and 333 working bus lines operating both ways, which results in 25849 bus runs in a single day.

In order evaluate the performance of the different approaches we first build visit plan requests. Using two of the 75 available POIs as the starting and ending points of the tour, we produce $75 \times 75 = 5625$ requests for visit plans. Note that this also includes the case in which the same POI is used as both starting and ending point. This simply means that a route would start and end at the same position, but it might still contain an arbitrary number of POIs to visit based on the user's specified time budget. The time budget is set as either 4, 6 or 8 hours. Starting times of the visit plan are either 10:00 or 12:00. This gives us 6 time-spans, 10:00 to 14:00 (4 hours), 10:00 to 16:00 (6 hours), 10:00 to 18:00 (8 hours), 12:00 to 16:00 (4 hours), etc. In addition, we also consider 5 different user profiles. To summarize, we have $5 \times 5625 \times 6 = 168,750$ requests for visit plans. The following approaches are compared:

(1) AvgILS. This is the approach implemented by García et al. [6]. Their ILS approach is computed using average travel times. Note that this approach is validated against our route planner to assess how many infeasible plans are produced. The evaluation scores for infeasible plans are counted as 0.

(2) RepAvgILS. AvgILS combined with the repairing method proposed by García et al. [6] which is described in section 5.

Our three designed approaches, **(3) SILS**, **(4) TRILS**, and **(5) PHILS**, which dynamically adjust the visit plan according to the real travel times provided by a route planner.

Each of the approaches is evaluated under two different circumstances: **Without Delays (ND)**, i.e. the case in which all bus units run perfectly on time according to their timetable, and **With Delays (D)**, in which delays are simulated at the level of single transportation units. Note that we assume that the time of request is the start of each time-span, i.e. either 10:00 or 12:00. The delays are therefore introduced before the time of request.

The evaluated metrics are shown at the end of the results table. All experiments were conducted on a single machine with 40 GB of RAM and a 64 bit Intel Xeon E5-2640, 2.5 GHz processor. The route planner ran for the entire duration of the experiments.

6.2 Interpretation of results

Tables 1 to 5 show the results of our experiments for each approach without and with delays in the transportation network. Each table's cell condenses the scores obtained for all requests and the five considered profiles in the given timespan. Moreover, for each timespan and metric the best scores achieved are shown in red.

Without delays (left tables). SILS produced 0.07% more profit than RepAvgILS, while TRILS and PHILS obtain 0.05% and 0.06%, respectively. A first observation is that the differences between overall total scores (TS) increase the more infeasible plans are produced (VTW) by AvgILS. The reason is that our approaches produce only feasible plans and therefore manage to gain additional profit in these cases (infeasible solutions contribute zero profit to the TS). This is also reflected in the columns TRS and ARS, the total and average repaired scores for those infeasible solutions. For the infeasible plans the improvement with respect to RepAvgILS is as follows: SILS achieves 6.76% improvement whereas PHILS 6.06%, and TRILS 5.12%.

With delays (right tables). Interestingly, all approaches generate trip plans with higher scores because the number of infeasible trip plans decreases. The reason for this is that delays are simulated for randomly picked units under independent assumptions. Therefore when a user travels to a POI more options, namely the delayed units, are available to reach it, which might reduce the travel time. PHILS performs the best, with an improvement in the overall score of 0.05% wrt. RepAvgILS. This is followed by SILS (0.04%) and TRILS (0.035%). For the infeasible plans the improvement with respect to RepAvgILS is as follows: PHILS achieves a 6.9% improvement whereas SILS 6.6%, and TRILS 4.95%. This shows the robustness of our approaches in the presence of delays.

As expected RepAvgILS was the fastest ILS-based heuristic due to its simple repairing strategy, whereas PHILS required the longest time (26.12% slower) to find the final solution. However, the execution times of both SILS and TRILS, approx. 1.2% and 1.5% slower, respectively, are very close to that of RepAvgILS. In any case our approaches are able to generate a plan in under 15ms (on average) which is a nice achievement considering this includes the adjustment time and interaction with the route planner.

Conclusion. SILS, TRILS and PHILS manage to produce only feasible plans thanks to the novel interactive aspect, i.e. the two-way flow of information between the core approach, which assembles the solution, and the route planner. Our results also show that more infeasible plans are produced when the time budget is large or the visit ends late (visits are closer to the closing times of all POIs).

7 CONCLUSION AND FUTURE WORK

We presented, SILS, TRILS and PHILS, three novel approaches to solve the *tourist trip design problem with travel instructions* (TTDP-TI). This problem is an extension of the TTDP which, in addition

[4]General Directorate of ESHOT. http://www.eshot.gov.tr

	AvgILS without delays											AvgILS with delays										
	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB
10:00-14:00	1429877,3	50,84	8,57	-	-	6,52	1,24	5,72	2,24	2	12908	**1429979,6**	**50,84**	8,57	-	-	6,52	1,24	5,82	2,12	0	12629
10:00-16:00	1893186,9	67,31	7,98	-	-	8,73	1,24	9,64	2,45	97	6292	1897425,1	67,47	7,98	-	-	8,73	1,24	9,3	2,58	46	6010
10:00-18:00	2335471,4	83,04	7,72	-	-	10,81	1,23	13,65	3,84	330	3793	2345451,9	83,39	7,73	-	-	10,81	1,23	13,58	3,88	209	3657
12:00-16:00	1428277,3	50,78	8,76	-	-	6,39	1,2	5,73	1,66	108	12675	1430924,7	50,88	8,76	-	-	6,39	1,2	5,76	1,68	62	12453
12:00-18:00	1872254,5	66,57	7,81	-	-	8,56	1,16	9,23	2,29	352	6011	1881889,1	66,91	7,8	-	-	8,56	1,16	9,25	2,3	224	5852
12:00-20:00	2268541,3	80,66	7,34	-	-	10,48	1,17	12,38	3,44	689	3651	2284818,9	81,24	7,34	-	-	10,48	1,17	11,5	3,34	507	3523

Table 1: Results of the AvgILS experiments

	RepAvgILS without delays											RepAvgILS with delays										
	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB
10:00-14:00	**1429960,6**	**50,84**	8,57	**83,3**	**41,65**	6,52	1,24	5,72	2,24	0	12910	**1429979,6**	**50,84**	8,57	0	0	6,52	1,24	5,82	2,12	0	12629
10:00-16:00	**1901084,4**	**67,59**	7,98	**7897,5**	**81,42**	8,73	1,24	9,61	2,49	0	6323	**1901546**	**67,61**	7,97	**4120,9**	**89,58**	8,73	1,24	9,29	2,6	0	6024
10:00-18:00	2363974,4	84,05	7,74	28503	86,37	10,8	1,23	13,51	4,03	0	3837	2365012,7	84,09	7,71	19560,8	93,59	10,8	1,23	13,49	4,01	0	3664
12:00-16:00	1433242,4	50,96	8,77	4965,1	45,97	6,38	1,2	5,71	1,67	0	12727	1433662	50,97	8,76	2737,3	44,15	6,38	1,2	5,76	1,68	0	12476
12:00-18:00	1895734,8	67,4	7,81	23480,3	66,71	8,55	1,16	9,13	2,42	0	6096	1896849,3	67,44	7,8	14960,2	66,79	8,56	1,15	9,13	2,43	0	5911
12:00-20:00	2324103,1	82,63	7,44	55561,8	80,64	10,46	1,18	12,13	3,73	0	3714	2325649,1	82,69	7,38	40830,2	80,53	10,47	1,17	12,07	3,71	0	3578

Table 2: Results of the RepAvgILS experiments

	SILS without delays											SILS with delays										
	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB
10:00-14:00	1429926,3	50,84	8,57	49	24,5	6,52	1,24	5,72	2,24	0	12908	**1429979,6**	**50,84**	8,57	0	0	6,52	1,24	5,82	2,12	0	12629
10:00-16:00	1900967,6	67,59	7,98	7780,7	80,21	8,73	1,24	9,65	2,45	0	6320	1901444,9	67,61	7,98	4019,8	87,39	8,73	1,24	9,3	2,58	0	6020
10:00-18:00	**2365608,7**	**84,11**	7,71	**30137,3**	**91,33**	10,8	1,23	13,68	3,86	0	3883	2366033,7	84,13	7,7	20581,8	98,48	10,81	1,23	13,61	3,89	0	3677
12:00-16:00	**1433759,7**	**50,98**	8,76	**5482,4**	**50,76**	6,39	1,2	5,73	1,66	0	12747	**1433910,5**	**50,98**	8,76	**2985,8**	**48,16**	6,39	1,2	5,87	1,69	0	12487
12:00-18:00	**1897730,1**	**67,47**	7,79	**25475,6**	**72,37**	8,56	1,15	9,26	2,3	0	6135	**1898078,5**	**67,49**	7,79	**16189,4**	**72,27**	8,56	1,15	9,26	2,38	0	5907
12:00-20:00	2328254	82,78	7,32	59712,7	86,67	10,47	1,17	12,46	3,53	0	3793	2328678,3	82,8	7,31	43859,4	86,51	10,48	1,17	12,35	3,53	0	3619

Table 3: Results of the SILS experiments

	TRILS without delays											TRILS with delays										
	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB
10:00-14:00	1429926,3	50,84	8,57	49	24,5	6,52	1,24	5,72	2,24	0	12908	**1429979,6**	**50,84**	8,57	0	0	6,52	1,24	5,82	2,12	0	12629
10:00-16:00	1901007,6	67,59	7,98	7820,7	80,63	8,73	1,24	9,65	2,46	0	6322	1901460,7	67,61	7,98	4035,6	87,73	8,73	1,24	9,3	2,58	0	6018
10:00-18:00	2365043,9	84,09	7,72	29572,5	89,61	10,8	1,23	13,7	3,99	0	3915	2365623,1	84,11	7,71	20171,2	96,51	10,81	1,23	13,63	4,04	0	3709
12:00-16:00	1433735,6	50,98	8,76	5458,3	50,54	6,38	1,2	5,73	1,66	0	12756	1433906,3	50,98	8,76	2981,6	48,09	6,39	1,2	5,76	1,69	0	12496
12:00-18:00	1897355,7	67,46	7,79	25101,2	71,31	8,56	1,15	9,28	2,44	0	6160	1897846,3	67,48	7,79	15957,2	71,24	8,56	1,15	9,27	2,41	0	5941
12:00-20:00	2327203,1	82,74	7,34	58661,8	85,14	10,47	1,17	12,57	4,29	0	3896	2327957,7	82,77	7,32	43138,8	85,09	10,47	1,17	12,52	4,25	0	3702

Table 4: Results of the TRILS experiments

	PHILS without delays											PHILS with delays										
	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB	TS	AS	StD	TRS	ARS	ARP	StD	AET	StD	VTW	VTB
10:00-14:00	1429925,5	50,84	8,57	48,2	24,1	6,52	1,24	7,82	2,41	0	12910	**1429979,6**	**50,84**	8,57	0	0	6,52	1,24	8,03	2,42	0	12629
10:00-16:00	1900914,5	67,59	7,99	7727,6	79,66	8,73	1,24	12,07	2,95	0	6300	1901497,8	67,61	7,98	4087,64	88,86	8,73	1,24	12,02	2,77	0	6011
10:00-18:00	2365429,1	84,1	7,72	29957,7	90,78	10,8	1,23	16,59	4,44	0	3858	**2366359,4**	**84,14**	7,7	**20852,84**	**99,77**	10,81	1,23	16,87	4,42	0	3678
12:00-16:00	1432980,6	50,95	8,78	4703,3	43,5	6,38	1,21	7,8	1,94	0	12705	1433667,3	50,97	8,77	2742,6	44,23	6,38	1,2	7,99	2,12	0	12471
12:00-18:00	1897202,8	67,46	7,79	24948,3	70,87	8,56	1,15	11,21	2,97	0	6081	1897777,2	67,48	7,79	15888,1	70,92	8,56	1,16	11,2	2,87	0	5885
12:00-20:00	**2328961,9**	**82,81**	7,31	**60416,39**	**87,68**	10,48	1,17	14,9	4,31	0	3763	**2329146**	**82,81**	7,31	**44322,9**	**87,42**	10,48	1,17	14,46	4,32	0	3598

Table 5: Results of the PHILS experiments

METRICS: **Total Score (TS)**. Overall score for all visit plans requests. **Average Score (AS)**. The average score obtained, i.e. $\frac{TS}{5625\times5}$. **Total Repair Score (TRS)**. Overall score for the repaired visit plans. **Average Repair Score (ARS)**. The average score obtained from the repaired visit plans, i.e. $\frac{TRS}{VTW_{AvgILS}}$. **Avg. number of Recommended POIs (ARP).** Number of recommended POIs on average for the feasible visit plans. **Avg. Execution time (AET)**. The execution time in milliseconds to return a solution. Standard deviation from AS, ARP and AET are provided next to each corresponding column. **Number of plans with at least one violation of a POI's time window (VTW)**. Number of requests which have at least one visiting time out of the POI time window. **Number of plans which violate the time budget (VTB)**. After adjusting the plan using the route planner, some visits are shifted back and forth leading to violations of the time budge constraint t_{max}.

to producing a visit plan (i.e. the sequence of POIs to visit in a city), also produces a travel plan with instructions on how to reach those attractions using public transportation. The novelty of these approaches lies in the way the visit plan is dynamically adjusted according to real travel times. While average travel times are still used in the computation for efficiency, the solution is eventually adjusted with the information provided by a route planner. If the adjustment leads to an infeasible plan each approach takes different countermeasures to repair it. The search for an optimal solution continues on top of the repaired plan. This makes it possible to not only return feasible plans (without violations of the POI's time windows) without sacrificing profit, but also to return both visit and travel times in real-time. Moreover, our state-of-the-art route planner is able to model a large variety of events related to public transportation, such as walking times (between stations, to a station to take a bus, etc.), changing vehicles (transfers), but especially to model delays of transportation units. To the best of our knowledge, no previous approach was able to provide travel instructions for the visit plans at this level of realism.

As we showed in our experiments, SILS, TRILS and PHILS are at comparable performance levels in terms of collected profit, while all approaches manage to outperform ILS-heuristics based on estimated travel times even after a repair. Moreover even PHILS, which required the longest execution time, is able to produce plans in real-time.

In the future we would like to extend our approach by modeling further constraints and events. In addition we would like to focus more on the personalization aspect of the TRS. Finally, our approaches could be used to model dynamic changes, too, thanks to the ability to deliver fast plans. If the user deviates from the visit plan, e.g. if she enjoys staying at one place and prolongs the visit, a new plan has to be recomputed together with the travel plans. Our approaches seem to be a good fit for this problem too.

REFERENCES

[1] Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. 2010. Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I (ESA'10)*. Springer-Verlag, Berlin, Heidelberg, 12. http://dl.acm.org/citation.cfm?id=1888935.1888969

[2] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. 2016. *Route Planning in Transportation Networks*. Springer International Publishing, Cham, Switzerland. http://dx.doi.org/10.1007/978-3-319-49487-6_2

[3] Hannah Bast, Jonas Sternisko, and Sabine Storandt. 2013. Delay-robustness of transfer patterns in public transportation route planning. In *ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013*, Vol. 33. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[4] Pierpaolo Di Bitonto, Francesco Di Tria, Maria Laterza, Teresa Roselli, Veronica Rossano, and Filippo Tangorra. 2010. Automated Generation of Itineraries in Recommender Systems for Tourism. In *Current Trends in Web Engineering - 10th International Conference on Web Engineering, ICWE 2010 Workshops, Vienna, Austria, July 2010, Revised Selected Papers*. https://doi.org/10.1007/978-3-642-16985-4_48

[5] H. Ding, L. Ke, and Z. Geng. 2016. Route planning in a new tourist recommender system: A fireworks algorithm based approach. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. https://doi.org/10.1109/CEC.2016.7744300

[6] Ander García, Pieter Vansteenwegen, Olatz Arbelaitz, Wouter Souffriau, and María Teresa Linaza. 2013. Integrating public transportation in personalised electronic tourist guides. *Computers & OR* 40, 3 (2013). https://doi.org/10.1016/j.cor.2011.03.020

[7] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, and Nikolaos Vathis. 2017. Scenic route planning for tourists. *Personal and Ubiquitous Computing* 21, 1 (2017). https://doi.org/10.1007/s00779-016-0971-3

[8] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati E. Pantziou, Nikolaos Vathis, and Christos D. Zaroliagis. 2015. The eCOMPASS multimodal tourist tour planner. *Expert Syst. Appl.* 42, 21 (2015), 7303–7316. https://doi.org/10.1016/j.eswa.2015.05.046

[9] Damianos Gavalas, Michael Kenteris, Charalampos Konstantopoulos, and Grammati E. Pantziou. 2012. Web application for recommending personalised mobile tourist routes. *IET Software* 6, 4 (2012). https://doi.org/10.1049/iet-sen.2011.0156

[10] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati E. Pantziou. 2014. Mobile recommender systems in tourism. *J. Network and Computer Applications* 39 (2014). https://doi.org/10.1016/j.jnca.2013.04.006

[11] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati E. Pantziou, and Nikolaos Vathis. 2014. Efficient Heuristics for the Time Dependent Team Orienteering Problem with Time Windows. In *Applied Algorithms - First International Conference, ICAA 2014, Kolkata, India, 2014. Proceedings*. https://doi.org/10.1007/978-3-319-04126-1_13

[12] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. 2015. An Iterated Local Search Algorithm for Solving the Orienteering Problem with Time Windows. In *Evolutionary Computation in Combinatorial Optimization - 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, 2015, Proceedings*. https://doi.org/10.1007/978-3-319-16468-7_6

[13] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. 2015. SAILS: hybrid algorithm for the team orienteering problem with time windows. In *Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA)*.

[14] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255, 2 (2016). https://doi.org/10.1016/j.ejor.2016.04.059

[15] Gilbert Laporte, François V. Louveaux, and Hélène Mercure. 1992. The Vehicle Routing Problem with Stochastic Travel Times. *Transportation Science* 26, 3 (1992), 161–170. https://doi.org/10.1287/trsc.26.3.161

[16] Shih-Wei Lin and Vincent F. Yu. 2012. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research* 217, 1 (2012). https://doi.org/10.1016/j.ejor.2011.08.024

[17] Wouter Souffriau, Joris Maervoet, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. 2009. A Mobile Tourist Decision Support System for Small Footprint Devices. In *Bio-Inspired Systems: Computational and Ambient Intelligence, 10th International Work-Conference on Artificial Neural Networks, IWANN 2009, Salamanca, Spain, 2009. Proceedings, Part I*. https://doi.org/10.1007/978-3-642-02478-8_156

[18] Kadri Sylejmani, Jürgen Dorn, and Nysret Musliu. 2012. A Tabu Search approach for Multi Constrained Team Orienteering Problem and its application in touristic trip planning. In *12th International Conference on Hybrid Intelligent Systems, HIS 2012, Pune, India, 2012*. https://doi.org/10.1109/HIS.2012.6421351

[19] Kadri Sylejmani, Jürgen Dorn, and Nysret Musliu. 2017. Planning the trip itinerary for tourist groups. *Information Technology & Tourism* (2017). https://doi.org/10.1007/s40558-017-0080-9

[20] Gytis Tumas and Francesco Ricci. 2009. Personalized Mobile City Transport Advisory System. In *Information and Communication Technologies in Tourism, ENTER 2009, Proceedings of the International Conference in Amsterdam, The Netherlands, 2009*. https://doi.org/10.1007/978-3-211-93971-0_15

[21] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. 2009. Iterated local search for the team orienteering problem with time windows. *Computers & OR* 36, 12 (2009). https://doi.org/10.1016/j.cor.2009.03.008

[22] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. 2011. The City Trip Planner: An expert system for tourists. *Expert Syst. Appl.* 38, 6 (2011). https://doi.org/10.1016/j.eswa.2010.11.085

[23] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10. https://doi.org/10.1016/j.ejor.2010.03.045

[24] C. Verbeeck, Kenneth Sörensen, El-Houssaine Aghezzaf, and Pieter Vansteenwegen. 2014. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research* 236, 2 (2014). https://doi.org/10.1016/j.ejor.2013.11.038

[25] Cédric Verbeeck, Pieter Vansteenwegen, and El-Houssaine Aghezzaf. 2017. The time-dependent orienteering problem with time windows: a fast ant colony system. *Annals of Operations Research* (2017). https://doi.org/10.1007/s10479-017-2409-3

[26] Wolfgang Wörndl, Alexander Hefele, and Daniel Herzog. 2017. Recommending a sequence of interesting places for tourist trips. *J. of IT & Tourism* 17, 1 (2017). https://doi.org/10.1007/s40558-017-0076-5

Context-Aware Tourist Trip Recommendations

Christopher Laß
Department of Informatics
Technical University of Munich
Boltzmannstr. 3
85748 Garching bei München, Germany
christopher.lass@tum.de

Daniel Herzog
Department of Informatics
Technical University of Munich
Boltzmannstr. 3
85748 Garching bei München, Germany
herzogd@in.tum.de

Wolfgang Wörndl
Department of Informatics
Technical University of Munich
Boltzmannstr. 3
85748 Garching bei München, Germany
woerndl@in.tum.de

ABSTRACT

Mobile and web-based services solving common tourist trip design problems are available, but only few solutions consider context for the recommendation of point of interest (POI) sequences. In this paper, we present a novel approach to incorporating context into a tourist trip recommendation algorithm. In addition to traditional context factors in tourism, such as location, weather or opening hours, we focus on two context factors that are highly relevant when recommending a sequence of POIs: *time of the day* and *previously visited point of interest*. We conducted an online questionnaire to determine the influence of the context factors on the user's decision of visiting a POI and the ratings of the POIs under these conditions. We integrated our approach into a web application recommending context-aware tourist trips across the world. In a user study, we verified the results of our novel approach as well as the application's usability. The study proves a high usability of our system and shows that our context-aware approach outperforms a baseline algorithm.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Context-Aware Recommender System, Tourist Trip Recommendation, Point of Interest, Tourism

1 INTRODUCTION

Recommender Systems (RSs) are commonly known as software systems that suggest certain items to users in a predictive manner [1]. These systems facilitate the presentation of the available information typically by comparing user preferences to some reference attributes. However, RSs can deliver more sophisticated suggestions by adapting to the specific contextual situation of the recommendation. Hence, context-aware recommender systems (CARSs) provide different movie suggestions based on contextual factors like a user's mood or the time of the day.

In the tourism domain, CARSs are also being developed and researched. Several relevant contextual factors (e.g., weather) and their respective contextual conditions (e.g., raining) have already been identified. For example, a CARS reduces the relevance of outdoor activities while it is raining [3].

Most tourism CARSs focus on suggesting single points of interest (POIs). Only few solve route-planning problems for tourists who want to visit multiple interesting POIs consecutively. This problem statement is summarized as the Tourist Trip Design Problem (TTDP) [13]. The TTDP defines the generic problem of personalized tourist trip generation and is commonly seen as an extension of the Orienteering Problem (OP) [23]. The basic idea is to maximize an objective score between an specific start and end point with several POIs in between [11].

In previous works, we aimed to solve the TTDP and introduced a mobile application and web service for tourist trip recommendations around the world. It takes the user's preferences, time and budget into account [16]. However, contextual information was not considered. In this work, we propose a novel, context-aware route recommendation algorithm that enhances our previous and related work. It incorporates various contextual information, including two that are especially relevant for POI sequences.

The rest of the paper is organized as follows. In Section 2 we present context factors that our CARS observes and explain how the respective contextual conditions ratings have been acquired. In Section 3, our novel context-aware route recommendation algorithm is presented. Section 4 describes the application *TourRec*, implementing our algorithm. In Section 5, the introduced CARS is evaluated against the RS from our previous work. Section 6 and 7 list related work and conclude the paper.

2 EVALUATING CONTEXT FACTORS FOR TOURIST TRIPS

In this section we briefly discuss context and explain how context is relevant for our RS. In order to integrate context-aware information into a tourist trip RS, we first have to identify appropriate context factors for the tourism domain and assess the influence of selected context factors. Also, the effects of each context factor under several contextual conditions on the user's route satisfaction have to be investigated. Therefore, we conducted an online study to observe context factors specifically relevant for sequences of POIs and derive information of similar pre-existing research for other context factors.

2.1 Context in Tourism Recommender Systems

A common definition describes context as "any information that can be used to characterize the situation of a [...] person, place, or object." [12]. Since we are only interested in information that is relevant in the tourism domain, we limit and categorize "any information" to physical context. Physical context can be described as the user's immediate physical surroundings. This includes, but is not limited to, *time of the day*, *light*, *weather*, *date*, *season*, and *temperature* [8]. This information could be retrieved by modern smartphones with a GPS sensor or light sensor in conjunction with

the current time. However, this does not work well for predictions. For a route RS this data mostly has to be retrieved by external services such as openweathermap[1]. Furthermore, the system itself has to be aware of the users physical context at each segment of the route recommendation. For example, a recommended route should contain less outdoor activities during the night or while it is raining.

Another relevant type of context which we do not yet consider in this work is social context. Social context can be described as the user's social group composition at the time of taking the recommendation. The user's standing and role in the group is also an important factor [2]. For example, a recommended route should contain no nightlife activities such as going to a club when children are part of the group.

2.2 Acquiring Context Relevance

We designed an online questionnaire to acquire quantitative measures of how selected contextual factors influence a user's decision of going to a POI. The following approach assesses the context relevance and is based on a methodology presented by Baltrunas et al. [3].

For the preliminary questionnaire a set of possible context factors should be selected by domain experts. The questionnaire participants are asked to imagine certain conditions and whether a specific context factor (e.g., weather) has a positive or negative influence on the rating of a particular item [1, 2].

With this methodology we observe the context factors *time of the day* and *previously visited POI*. These are especially crucial for sequences of POIs and have not yet been observed in related work. For other context factors like *day of the week*, *weather* and *temperature*, which are also relevant for single POIs, we can rely on [3]. Also *opening hours* is considered, which is a context factor that does not require a preliminary user study. The mentioned context factors are later incorporated within the context-aware recommendation algorithm.

Preliminary, twelve POIs in Munich, Germany have been selected and mapped into six predefined categories: *Arts and Museum*, *Food*, *Music Event*, *Nightlife Spot*, *Outdoors and Recreation* and *Shopping*. It is assumed that categories represent all their corresponding POIs. Figure 1 shows how participants are asked whether they would visit a certain POI just after they have been to a different POI. Additionally, we asked the participants at which times they would go to certain POIs.

The aim of this study was to evaluate the influence of the selected context factors on their decisions to visit a category represented by a selected POI as well as the change of POI popularity precipitated by contextual conditions. In total, we received 324 responses by 27 participants.

The measured relevance (U) for each context factor for all POI categories are computed and listed in Table 1. It is normalized to an interval $[0, 1]$; where $U = 0$ means that the context factor does not have any influence for this POI category. U is also relevant for the actual context-aware route recommendation algorithm and is there being utilized as a weighting factor for the context assessment in Equation 5.

[1]https://openweathermap.org/

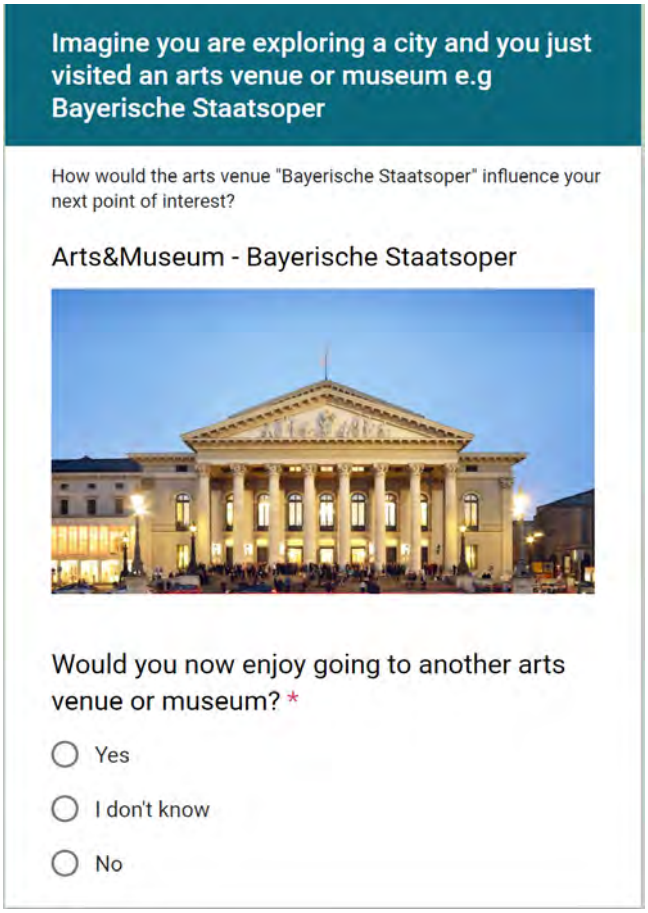


Figure 1: Online questionnaire to acquire context relevance.

In addition to the measured relevance (U) of a context factor, our context-aware approach (cf. subsection 3.3) is also dependent on ratings for POIs under different contextual conditions. The dataset resulting from the previous conducted questionnaire can also be utilized to determine such a rating. To make the responses quantifiable *Yes*, *I don't know* and *No* are mapped to the values $2, 1, 0$. A simple approach would be to use the mathematical expectation value as a rating of a POI category for each contextual condition. However, this does not respect the variation of the rating for a POI when a contextual condition holds or not. Informally speaking, if a POI category is typically very popular, except during *night*, the expectation value would not reflect the real value of the contextual condition *night*. For example, the expectation value for the category *food* is 1.3. However, if one only considers ratings for *food* under the contextual condition *night*, the expectation value is 0.749. Hence, we must present a comparison between the average ratings of POI and ratings of the same items assuming a certain contextual condition holds. We achieve this by dividing the expected value for a specific contextual condition by the expected value over all ratings for this POI category. For the category *Food* during *night* time, the calculation is therefore: $0.749/1.3 = 0.58$. All computed

Table 1: Measured relevance of the contextual factors by POI categories.

Contextual Factor	Arts and Museum	Music Event	Nightlife Spot	Food	Outdoors and Recreation	Shopping
Previously visited POI	0.52	0.31	0.26	0.49	0.33	0.42
Time of the day	0.48	0.69	0.74	0.51	0.67	0.58

ratings for POI categories in different contextual conditions are displayed in Table 2.

3 A NOVEL APPROACH FOR CONTEXT-AWARE ROUTE RECOMMENDATIONS

This section gives a detailed explanation of paradigms for incorporating context into RSs, the baseline path-finding algorithm and our approach for a context-aware path-finding algorithm.

3.1 Paradigms for Incorporating Context in Recommender Systems

This section describes how RS and CARS can be modeled and which paradigms exist to integrate context into a traditional, two-dimensional (2D) RS. 2D RSs try to estimate the rating function R by considering only the *User* and *Item* dimensions [2]:

$$R : User \times Item \rightarrow Rating \tag{1}$$

This rating function can be extended to model a three dimensional (3D) recommendation [2]:

$$R : User \times Item \times Context \rightarrow Rating \tag{2}$$

Adding context in the rating function increases the complexity of the recommendation algorithm. This is a non-trivial problem. Adomavicius and Tuzhilin [2] identify three different paradigms how to incorporate context in a traditional, 2D recommendation process:

Contextual pre-filtering *(or contextualization of recommendation input)*: The context is utilized to construct a dataset only with the most relevant data. After that, a traditional RS can generate the actual recommendations.

Contextual post-filtering *(or contextualization of recommendation output)*: In contrast to contextual pre-filtering, the traditional RS is executed on the *entire* data first and afterwards the context is applied on the resulting set. This can be achieved by:

- *Filtering* out recommendations that are irrelevant (in a given context), or
- *Adjusting* the ranking of recommendations on the list (based on a given context).

Contextual modeling *(or contextualization of recommendation function)*: In this paradigm the 2D RS must be modified and directly incorporate context into the recommendation algorithm.

3.2 Baseline Algorithm

We have improved a route recommendation algorithm in previous work [16, 24] which is not context-aware.

The general idea is to combine as many single POIs as possible to maximize the entertainment for the user while still respecting existing constraints like time. The process of generating a path from an origin to a destination while suggesting relevant POIs in between can be generally divided into two subtasks:

- The POI gathering and scoring, and
- executing a path-finding algorithm to find the optimal route consisting of a subset of the gathered POIs.

For the POI gathering, we are using the Foursquare API[2] to search for POIs in the general area between the source and the destination point. The gathered items are classified into six categories, users can give preferences for these categories (see below). Our algorithm then computes a score for each item. The score is based on the total Foursquare rating and the number of votes, and also the user preference for the corresponding category [24].

The algorithm to combine the POIs to a reasonable route is based on the well-known Dijkstra's algorithm to find the shortest path in a graph. Dijkstra's algorithm is an iterative algorithm that provides the shortest path from one particular starting node to all other nodes in a graph with non-negative edge path costs. In our scenario, the nodes are the places with the associated score and the edges represent the distance between the places.

Prior to the graph spanning, each POI is assigned with a value for the time to spend there. Then, a weighted graph is created using an feasible time value to walk the direct physical distance between each vertex as edge weight. Prior to the comparison of a subpath with another path, it is checked whether the subpath exceeds the specified timeframe. If the timeframe is exceeded, the subpath will be rejected. If another valid subpath from the origin to the immediate vertex can be found prior to the current path, they are compared against each other.

To generate not the shortest, but the best path in the POI graph, we maximize the fraction entertainment/distance for each subpath. The entertainment value is the accumulated sum of the scores of all items on the subpath. To adapt the number of items per category, the baseline algorithm uses the following formula Equation 3.

$$S = p_{pref,poiCategoriesInPath} \times entertainment \tag{3}$$

The idea is to maximize the product of entertainment and Pearson's correlation coefficient between the user's preferences and the amount of POIs per category in the observed path. Pearson's coefficient p gives values between -1 (indicating perfect negative correlation) and +1 (perfect correlation), with 0 meaning no correlation exists between the datasets. Using the correlation coefficient

[2]https://developer.foursquare.com/start/search

Table 2: Ratings for points of interest categories in different contextual conditions.

Contextual Condition\POI Category	Arts and Museum	Music Event	Nightlife Spot	Food	Outdoors and Recreation	Shopping
Previously visited POI (category)						
Arts and Museum	1.36	1	1.16	1.43	1.25	0.72
Food	1.4	1.06	1.77	0.19	1.28	1.18
Music Event	0.04	1.32	1.69	1.1	0.6	0.11
Nightlife Spot	0	1.45	1.43	1.04	0.13	0
Outdoors and Recreation	1.63	1.42	0.86	1.37	0.76	1.52
Shopping	0.91	0.52	0.79	1.45	0.97	1.25
Time of the day						
Morning	1.56	0.1	0.19	0.3	1.36	1.82
Midday	1.56	0.19	0.07	1.29	1.41	1.78
Afternoon	1.48	0.68	0.15	0.85	1.41	1.71
Evening	0.64	1.71	0.79	1.4	0.76	0.8
Night	0.42	1.55	2	0.58	1.07	0.11

aims to balance the amount of POIs in each category more appropriate in relation to the user's preferences. The extension of this adapted POI score with context-awareness is explained in the following subsection.

3.3 Incorporating Context into the Baseline Algorithm

The first challenge that arises is to determine how context-awareness can be calculated for a route. Our collected dataset includes two indications that can be utilizes for this task. First, ratings for categories in different contextual conditions as displayed in Table 2. A rating $r_{TC1...Ck}$ indicates the evaluation for the POI category T made in the context $C1, ..., Ck$ and must be in the interval $[0, 2]$. Second, the relevance of contextual factors $U_{C1...Ck}$ of each context $C1, ..., Ck$ on a POI category T as displayed in Table 1. Like illustrated in subsection 2.2 the measured relevance must be in an interval between $[0, 1]$.

Given this data we can calculate a context-awareness factor $\overline{C}$ with a simple weighted arithmetic mean:

$$\overline{C} = \frac{\sum_{i=1}^{k} U_{Ci} r_{TCi}}{\sum_{i=1}^{k} U_{Ci}} \tag{4}$$

$\overline{C}$ can now be used to extend the 2D recommender baseline algorithm by scaling the result of its comparison function:

$$S = p_{pref, poiCategoriesInPath} \times entertainment \times \overline{C} \tag{5}$$

According to the given constraints, also $\overline{C}$ is in the interval $[0, 2]$ with 0 essentially nulling the score S while 2 would double its value.

To better explain the methodology we can illustrate it with an example comparison considering the two context factors *time of the day* and *previously visited POI*:

It is 5 pm and the user has just been to a restaurant. The CARS should now calculate the score for another restaurant. In this scenario the 2D comparison algorithm would calculate a score of 9.5. The context-aware comparison algorithm (Equation 5) extends the 2D comparison algorithm (Equation 3) by the context-awareness factor $\overline{C}$. To calculate $\overline{C}$, we use the values 0.49 and 0.51 for relevance of the context factors from Table 1 and the values 0.19 and 0.85 for ratings for the category *food* in the current contextual condition from Table 2. After calculating $\overline{C}$, the 2D score of 9.5 is downscaled to 5.

$$\overline{C} = 0.526 = \frac{0.19 \times 0.49 + 0.85 \times 0.51}{0.49 + 0.51} \tag{6}$$

$$S = 5 = 9.5 \times 0.526 \tag{7}$$

According to 3.1, one could assume that this algorithm adheres to *contextual post-filtering*. However, the definition explicitly states, that the traditional RS must be executed on the *entire* data first. Since this is not the case, the paradigm *contextual modeling* was utilized to incorporate context into the baseline.

The full set of context factors considered in the current implementation and their values (contextual conditions) are:

- Previously visited POI (category): arts and museum, food, music event, nightlife spot, outdoors and recreation, shopping
- Time of the day: morning (8am - 12pm), midday (12pm - 2pm), afternoon (2pm - 6pm), evening (6pm - 10pm), night (past 10pm)
- Day of the week: working day, weekend
- Weather: sunny, cloudy, clear sky, rainy, snowing
- Temperature: hot, warm, cold
- Opening hours: open, closed

One benefit of the weighted arithmetic mean is the independence of the number of context factors. This list can easily be extended. Also the amount of context factors applied on POIs within a path can vary. For example, designing context factors only known for a specific POI category, e.g. *nightlife spot*, is not a concern. On the other hand, one disadvantage resulting from considering multiple context factors for $\overline{C}$ is that a supposedly drastic condition, e.g. the POI is closed, can be balanced out by a different condition such as *sunshine*.

4 THE TOURREC WEB APPLICATION

This section firstly presents the multi-tiered and service-oriented system architecture we introduced in (removed for review process). One of the main goals was to facilitate the integration of additional data sources, path-finding algorithms and clients. We then present the user interfaces of our web application *TourRec* [3].

4.1 Architecture

The system is distributed across multiple physical devices offloading application logic, computation and storage onto multiple web services running in the cloud. The applications multi-tier architecture can be partitioned into the presentation tier, application logic tier and data tier, while the application logic tier itself is also partitioned. This architectural style decomposes an application into loosely coupled services, functionality can thereby be easily reused. For example, clients do not have to re-implement the whole application logic but rely on the application logic tier. Other advantages are an improved modularity and the fact that each segment is easier to understand, develop and test. It also simplifies further development since components can be assigned more precisely to experts in their respective fields. An iOS developer gets to develop the iOS app, the android developer the android application and the data scientists can improve the algorithm. Also multiple people can work parallel and independently on different components.

Additionally, the whole application has been containerized with Docker [4] containers. Containers consist of a complete and isolated run time environment: the software including all its dependencies, libraries and other binaries, and configuration files needed to run it. By containerizing our application, the differences in OS distributions and underlying infrastructure are abstracted away. This makes it fairly easy for possible new contributors to run the whole project on their local environment. In addition, it enables continuous delivery and deployment.

In the following, the three tiers are presented.

4.1.1 Presentation Tier. The web application being demonstrated as well as the mobile application we have developed in our previous work is representative for the presentation tier. It is end user facing, aimed to provide high user satisfaction and is responsible to handle user input and display computed information. It utilizes services from the underlying application logic tier and external services like Google Maps.

4.1.2 Application Logic Tier. The two main functionalities of this tier are the gathering of POIs and executing path-finding algorithms. First, POIs are gathered from multiple external service providers such as Foursquare[5] and normalized. Then, this data is passed over to the path-finding algorithms that are executed afterwards. Each path-finding algorithm is extracted into its own dedicated microservice and communicates with the application logic tier via HTTP in order to return a ordered list of POIs to visit. The path-finding microservices can be implemented using arbitrary programming languages, databases, hardware and software environment.

4.1.3 Data Tier. User feedback is being stored and handled within this tier. As it might not seem urgent to dedicate an own tier for this data, the need increases when user accounts are introduced, for example.

4.2 Interfaces

The web application is built with help of the javascript framework Vuejs[6] and the CSS framework Bulma[7]. It is mainly structured into three segments *search*, *recommendation*, *feedback*.

In the *search* segment, as seen in Figure 2, users can enter their preferences for all six predefined categories, the origin and destination as well as the time frame for the route. The input is validated both client side as well as server side. For example, the maximum distance between origin and destination is 7 kilometers and the maximum time frame is 12 hours.

The *recommendation* segment, as seen in Figure 3, is structured as follows. A map with the suggested POIs and a rendered walking path on it is located on the left-hand side. On the top left-hand side some contextual information that has been acquired by the system and is relevant for the user for this situation is displayed. Finally an ordered list of POIs and their respective estimated time of arrival and departure can be found beneath the contextual information.

The *feedback* segment gives the user a short introduction into the feedback process. In essence, it is a simple table with multiple statements which can each be answered with radio buttons on a five-point Likert scale ranging from *strongly disagree* to *strongly agree*. In section 5 we describe the feedback setup in greater detail. Note, that only one route per request gets displayed. The application logic tier randomly selects either the baseline or context-aware path-finding algorithm. The algorithm name is stored in conjunction with the feedback a user provides.

5 USER STUDY

The research question of this paper is whether a context-aware algorithm distinguishing between several contextual conditions can improve the trip recommendations generated by a baseline. We conducted a user study to evaluate the performances of both algorithms. We spread the link to the *TourRec* application via mail and added questionnaires to the interfaces which the users were asked to complete. The participants could access the application on any device with an internet connection since it is publicly available.

5.1 System Usability Score

The System Usability Scale (SUS) is a questionnaire for measuring how people perceive the usability of a computer system [6]. The questionnaire is composed of ten usability statements with five possible response options on a scale ranging from strongly disagree to strongly agree. SUS is technology independent and can be used for hardware, software, websites, mobile applications and more. The key benefits of SUS are reliability, validity, no need a baseline and the fact that it is an industry standard [7].

19 participants completed the SUS questionnaire after using *TourRec*, the average score was 84,167. With the help of Sauro's graph, the score approximately converts to a percentile rank of

[3] https://tourrec.arubacao.com/
[4] https://www.docker.com/
[5] https://foursquare.com/
[6] https://vuejs.org/
[7] http://bulma.io/

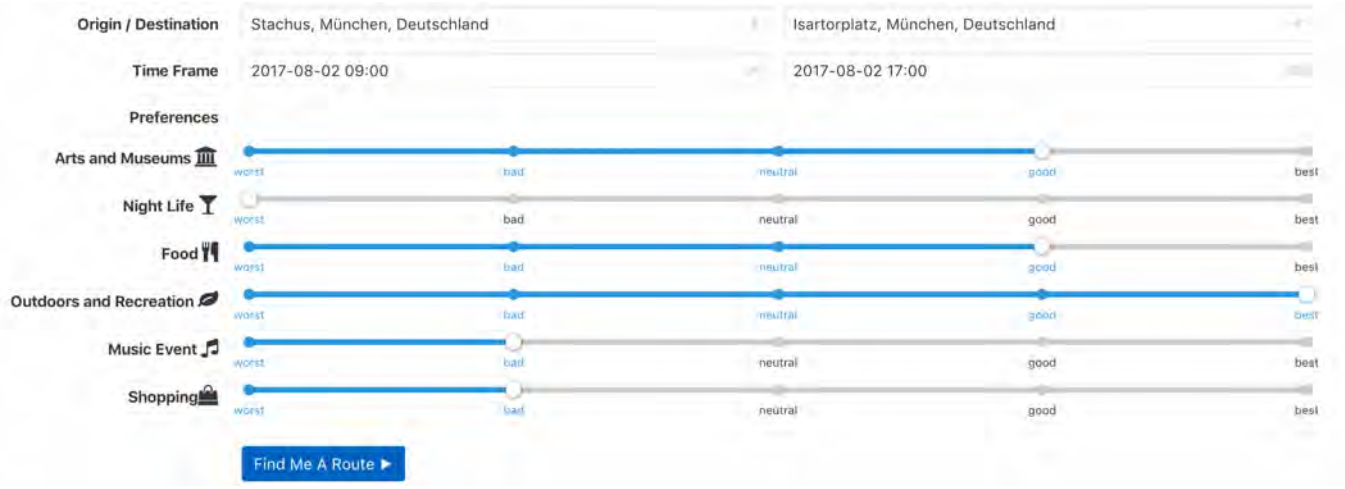
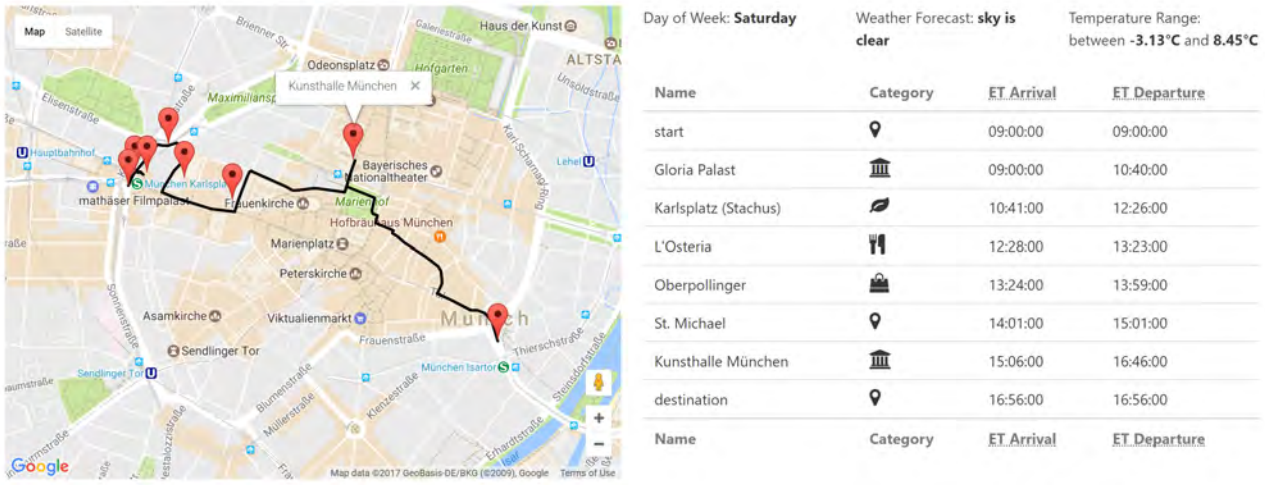


Figure 2: *TourRec* Search Interface



Name	Category	ET Arrival	ET Departure
start		09:00:00	09:00:00
Gloria Palast		09:00:00	10:40:00
Karlsplatz (Stachus)		10:41:00	12:26:00
L'Osteria		12:28:00	13:23:00
Oberpollinger		13:24:00	13:59:00
St. Michael		14:01:00	15:01:00
Kunsthalle München		15:06:00	16:46:00
destination		16:56:00	16:56:00
Name	Category	ET Arrival	ET Departure

Figure 3: *TourRec* Response Interface

94% [21]. This means *TourRec* performs better than about 94% of systems tested in terms of perceived usability. Everything about 90% can be interpreted as an A in school grades. However, since *TourRec* is accessible from virtually any device, the actual systems usability varies for different screen sizes, operating systems and browser vendors.

5.2 Algorithm Performance

In addition to the SUS, we conducted an A/B test to measure the effect of the novel approach on the user's route satisfaction compared to the baseline system that does not exploit context at all. Hence, only one tourist trip recommendation is displayed after every request. Apart from the route, users are not able to distinguish between them. The recommendation screen shown in Figure 3 displays contextual information (e.g. the weather) whether or not the context is actually considered.

After every recommendation, a questionnaire for this part of the evaluation is presented to the user. It is composed of the following six statements and also with five possible response options on a scale ranging from *strongly disagree* (1) to *strongly agree* (5):

(1) Overall, I am satisfied with the recommended tour
(2) The number of places in my route is well chosen
(3) The selection of different categories in the trip is satisfying
(4) Places are suggested at the right times during the tour
(5) The tour is feasible for a walking tourist
(6) I consider taking this route myself

In total, 15 forms were completed for the baseline algorithm and 9 for the context-aware approach. Figure 4 illustrates the performance of both algorithms for each of the six questions in subsection 5.2. Our novel approach for context-aware route recommendation performs somewhat better in the *overall satisfaction* ($\varnothing : 3,67, \sigma : 1,41$)

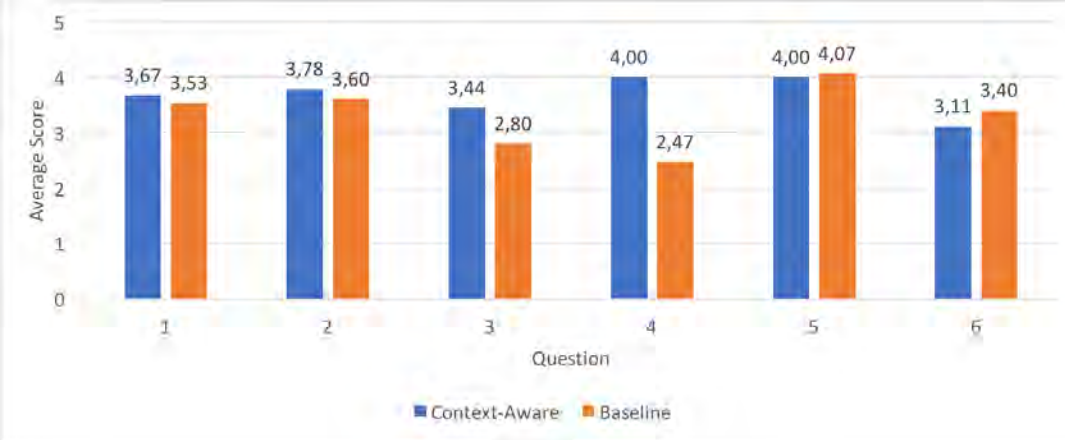


Figure 4: Algorithm Performance Results

and *number of places* ($\varnothing : 3,78, \sigma : 1,39$). In terms of *feasible walking route* and *consider taking the route* the context-aware algorithm is rated slightly lower than the baseline. However, for these four mentioned questions the actual difference is almost neglectable. The biggest difference can be seen for *diversity of categories* and especially *right times* where the context-aware algorithm outperforms the baseline.

A Mann-Whitney U test shows that the difference in *right times* is significant for $\alpha = 0.01$ while the other results we obtained are not yet significant. We conclude that our novel approach leads to improved recommendations but we have to conduct a larger user study in the future to verify our results.

6 RELATED WORK

Some applications solving the TTDP have been developed [9, 23] and numerous publications on this topic ranging from the avoidance of trafficked walking paths [20] to randomizing these [19] exist. In this section, we highlight some important related work based on a general overview of [18].

Gionis et al.[14] and Lim [17] consolidated POIs into categories to enforce a predefined visiting order. In [17] the visiting order was defined by user preferences, and time and budget constraints. Instead of enforcing a specific order, Vansteenwegen et al. [22] recommended tours comprising POI categories that best match user preferences while adhering to these trip constraints.

Tour recommendation can also be formulated as a Generalized Maximum Coverage Problem [4]. The objective here is to find an optimal set of POIs considering its rating and the user's preferences. Brilhante et al. then extended their algorithm later by incorporating a variation of the Travelling Salesman Problem to find the shortest path within an optimal set of POIs [5]. To get from POI to POI within a route, Kurashima et al. also consider different transport modes utilizing the Markov model depending on user preferences and frequently traveled routes [15]. With patterns derived from taxi GPS traces, Chen et al. developed a more context-aware solution considered traveling times based on different traffic conditions [10].

7 CONCLUSION

In this paper, we presented the web application *TourRec* that allows context-aware tour recommendations in arbitrary locations across the world. The system takes a starting point, a destination, a timeframe and user preferences for six predefined categories into account. It solves a variant of the OP applied to the tourism domain. In a preliminary questionnaire, the influence of the context factors *time of the day* and *previously visited POI* were measured as well as ratings for POI categories in different contextual conditions. The results are utilized within the context-aware algorithm.

Context-awareness is incorporated into the baseline algorithm as a scaling factor altering a POI's score depending on the immediate contextual condition. The focus and innovation of our work is on recommending sequences of items. Therefore the influence of an already visited POI on the score of additional items based on their category is important. Users may not be interested in another restaurant if they just had lunch or dinner, for example. In a user study we evaluated that the incorporation of context-awareness leads to a slightly improved user satisfaction and a significant improvement of recommending POIs at the right time.

One disadvantage of our approach is the possible equalizing of two or more extreme contextual conditions due to the weighted arithmetic mean. A modified version could solely consider the context factor that has the largest negative or positive effect on a POI. Using one single factor could potentially characterize the POI's score in a more user satisfying way. Another improvement can be achieved by increasing the number of predefined categories or introduce subcategories. The current limitation of only six categories has led to issues in the route recommendation scenario.

We have designed our framework and architecture for easy extension. We are working on a mobile solution [16] because context-aware route planning seems especially promising in a scenario of mobile users with smartphones visiting a city. In this case, the recommended items should be adapted to the current position and other contextual conditions of the user. We also plan to implement more path-finding algorithms and compare them with the approach presented in this paper in larger user studies. Additional future work includes recommending for groups of visitors.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.* 17, 6 (June 2005), 734–749. DOI: https://doi.org/10.1109/TKDE.2005.99

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2008. Context-aware Recommender Systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)*. ACM, New York, NY, USA, 335–336. DOI: https://doi.org/10.1145/1454008.1454068

[3] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. 2012. Context Relevance Assessment and Exploitation in Mobile Recommender Systems. *Personal Ubiquitous Comput.* 16, 5 (June 2012), 507–526. DOI: https://doi.org/10.1007/s00779-011-0417-x

[4] Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2013. Where Shall We Go Today?: Planning Touristic Tours with Tripbuilder. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 757–762. DOI: https://doi.org/10.1145/2505515.2505643

[5] Igo Ramalho Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2015. On planning sightseeing tours with TripBuilder. *Information Processing and Management* 51, 2 (2015), 1 – 15. DOI: https://doi.org/10.1016/j.ipm.2014.10.003

[6] John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[7] John Brooke. 2013. SUS: A Retrospective. *J. Usability Studies* 8, 2 (Feb. 2013), 29–40. http://dl.acm.org/citation.cfm?id=2817912.2817913

[8] P. J. Brown, J. D. Bovey, and Xian Chen. 1997. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications* 4, 5 (Oct 1997), 58–64. DOI: https://doi.org/10.1109/98.626984

[9] Luis Castillo, Eva Armengol, Eva OnaindÃŋa, Laura SebastiÃą, JesÃžs GonzÃąlez-Boticario, Antonio RodrÃŋguez, Susana FernÃąndez, Juan D. Arias, and Daniel Borrajo. 2008. samap: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications* 34, 2 (2008), 1318 – 1332. DOI: https://doi.org/10.1016/j.eswa.2006.12.029

[10] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu. 2015. TripPlanner: Personalized Trip Planning Leveraging Heterogeneous Crowdsourced Digital Footprints. *IEEE Transactions on Intelligent Transportation Systems* 16, 3 (June 2015), 1259–1273. DOI: https://doi.org/10.1109/TITS.2014.2357835

[11] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. 2010. Automatic Construction of Travel Itineraries Using Social Breadcrumbs. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT '10)*. ACM, New York, NY, USA, 35–44. DOI: https://doi.org/10.1145/1810617.1810626

[12] Anind K. Dey. 2001. Understanding and Using Context. *Personal Ubiquitous Comput.* 5, 1 (Jan. 2001), 4–7. DOI: https://doi.org/10.1007/s007790170019

[13] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. A Survey on Algorithmic Approaches for Solving Tourist Trip Design Problems. *Journal of Heuristics* 20, 3 (June 2014), 291–328. DOI: https://doi.org/10.1007/s10732-014-9242-5

[14] Aristides Gionis, Theodoros Lappas, Konstantinos Pelechrinis, and Evimaria Terzi. 2014. Customized Tour Recommendations in Urban Areas. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*. ACM, New York, NY, USA, 313–322. DOI: https://doi.org/10.1145/2556195.2559893

[15] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. 2010. Travel Route Recommendation Using Geotags in Photo Sharing Sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, New York, NY, USA, 579–588. DOI: https://doi.org/10.1145/1871437.1871513

[16] Christopher Laß, Wolfgang Wörndl, and Daniel Herzog. 2016. A Multi-Tier Web Service and Mobile Client for City Trip Recommendations. In *The 8th EAI International Conference on Mobile Computing, Applications and Services (MobiCASE)*. ACM. DOI: https://doi.org/10.4108/eai.30-11-2016.2267194

[17] Kwan Hui Lim. 2015. Recommending Tours and Places-of-Interest Based on User Interests from Geo-tagged Photos. In *Proceedings of the 2015 ACM SIGMOD on PhD Symposium (SIGMOD '15 PhD Symposium)*. ACM, New York, NY, USA, 33–38. DOI: https://doi.org/10.1145/2744680.2744693

[18] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Personalized Tour Recommendation Based on User Interests and Points of Interest Visit Durations. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1778–1784. http://dl.acm.org/citation.cfm?id=2832415.2832496

[19] Claudio Lucchese, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. 2012. How Random Walks Can Help Tourism. In *Proceedings of the 34th European Conference on Advances in Information Retrieval (ECIR'12)*. Springer-Verlag, Berlin, Heidelberg, 195–206. DOI: https://doi.org/10.1007/978-3-642-28997-2_17

[20] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. 2014. The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media (HT '14)*. ACM, New York, NY, USA, 116–125. DOI: https://doi.org/10.1145/2631775.2631799

[21] J. Sauro. 2011. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. CreateSpace Independent Publishing Platform. https://books.google.co.uk/books?id=BL0kKQEACAAJ

[22] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. 2011. The City Trip Planner. *Expert Syst. Appl.* 38, 6 (June 2011), 6540–6546. DOI: https://doi.org/10.1016/j.eswa.2010.11.085

[23] Pieter Vansteenwegen and Dirk Van Oudheusden. 2007. The Mobile Tourist Guide: An OR Opportunity. *OR Insight* 20, 3 (2007), 21–27. DOI: https://doi.org/10.1057/ori.2007.17

[24] Wolfgang Wörndl, Alexander Hefele, and Daniel Herzog. 2017. Recommending a sequence of interesting places for tourist trips. *Information Technology & Tourism* 17, 1 (2017), 31–54. DOI: https://doi.org/10.1007/s40558-017-0076-5

Towards the Recommendation of Personalised Activity Sequences in the Tourism Domain

Gunjan Kumar, Houssem Jerbi, and Michael P. O'Mahony
Insight Centre for Data Analytics,
School of Computer Science, University College Dublin, Ireland
{firstname.lastname}@insight-centre.org

ABSTRACT

In this paper we consider the problem of recommending sequences of activities to a user. The proposed approach leverages the order as well as the context associated with the user's past activity patterns to make recommendations. This work extends the general activity recommendation framework proposed in [16] to iteratively recommend the next sequence of activities to perform. We demonstrate the efficacy of our recommendation framework by applying it to the tourism domain and evaluations are performed using a real-world (checkin) dataset.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Decision support systems*; *Spatial-temporal systems*;

KEYWORDS

Sequence Recommendation, Recommender Systems, Activity Recommendation, Activity Timeline Matching

1 INTRODUCTION

Internet and digital technologies have significantly influenced the tourism sector in the last decade resulting in a steady growth in e-tourism [7]. Users now have easy access to vast amounts of information on the web which assists them to plan trips, make reservations, and purchase products etc. However, the number of available choices have increased so rapidly that it has become difficult to find the right information at the right time. Thus, recommender systems, which have found immense success in e-commerce, have the potential to play a crucial role in e-tourism by providing personalised and relevant content to users [5, 14, 24, 27].

To provide useful recommendations, it is essential to capture the behaviour and needs of users, which has been particularly challenging in e-tourism [26]. However, as digital technologies have now permeated our daily lives to a great extent, many aspects of our lives can now be easily recorded in digital format. For example, physical activities performed, locations visited and media consumed by users can be recorded using mobile devices [12]. Moreover, mobile personal assistants, such as Google Now and Microsoft Cortana, are capable of passively recording the digital activities of users. These recordings, which contain the activity patterns and preferences of users, can facilitate the development of personalised recommender systems capable of generating recommendations at the right time and in the right way for a given user and context [11, 31, 32].

In our previous work [15, 16], we proposed a generic activity recommendation framework to recommend the *next activity* to perform to a user. Our approach was applied successfully in the lifelogging and urban computing domains, where activities included socialising, eating, etc. and modes of transport, respectively. In this paper, we extend the activity recommendation framework to address the task of recommending a *sequence of activities* to the user. Moreover, we apply our framework to the tourism domain, where a recommended sequence of activities might be, for example, visiting a zoo, eating Italian food, and then listening to live music.

Our work is motivated by the assumption that people tend to repeat similar patterns of activities under similar circumstances [29]. Hence, in order to infer the next activities for a user, it is important to consider the activity patterns performed in the past. At the same time, the context surrounding these activities significantly affects the next activities the user performs. The importance of modelling context has been recognised in both tourism [6, 17, 18] as well as recommender systems research [1]. Context is particularly important in tourism as the user is predominantly mobile [10]. For example, features such as the time of day, location and weather can determine whether a user visits a particular amusement park in the city or not.

In recommender systems research, the task of recommending sequences is comparatively under-explored [13, 27]. However, there exists works, particularly for points of interest/itinerary (LBSN) [20, 30, 34] and music playlists [2, 4, 8, 22, 23, 27] recommendation, which address this task. A popular approach for modeling sequences has been Markov-based models [4] and all-k^{th}-order Markov models [3, 9, 25, 28]. However, in general, these approaches are not suitable for modelling sequences of activities with multiple features or context and are limited to the Markov assumption which does not apply in all cases [4]. An alternative hierarchical graph-based approach to capture sequences and geographical hierarchies in location trajectories is presented in [19]. This is further enhanced in [35] by modeling location popularity and user experiences to mine popular travel sequences across users in a non-personalised manner. Similarly, graph-based models have been used for collaborative itinerary recommendation [33]. However, these approaches do not capture the context information associated with user activities.

The key distinguishing characteristic of our work is that the model captures both the past activities of users, together with the context associated with these activities, in order to recommend the next sequence of activities for users to perform. The main contributions of this work can be summarised as follows:

- The extension of the generic activity recommendation framework in [15, 16] to recommend the next sequence of activities that should be performed by users. For this, an iterative, content-based recommendation approach is proposed, which takes the sequence as well as the features associated with

previous activity occurrences into consideration to build the recommendation model (Section 2);
- The application of our proposed algorithm to the tourism domain. Experiments using a location checkin dataset [21] demonstrate the efficacy of our approach in recommending sequences given a diverse variety of activities and user activity patterns (Section 3).

2 RECOMMENDATION APPROACH

In this section, we formulate the problem of recommending the next sequence of activities to a user. These activities can be, for example, eating Italian food, shopping at a bookstore, listening to live music, etc. The proposed content-based sequence recommendation algorithm leverages sequential patterns in a user's past activities as well as the contextual information (for example, time of day, location, weather, etc.) associated with each activity occurrence.

2.1 Problem Formulation

We introduced the concept of an *activity object* and an *activity timeline* in [15]. An *activity object*, ao_i, refers to a single occurrence of an activity and consists of a set of features, $ao_i = \{v_i^1, v_i^2, ..., v_i^m\}$, which describe the context surrounding that particular occurrence of the activity. For example, an activity object can refer to an instance of 'a visit to a zoo' (i.e. the *activity name*) with associated contextual features, such as *time of day, geo-location, weather, popularity of the location*, etc. An *activity timeline* (or *timeline* for short) for a user is then a chronological sequence of all activity objects performed by that user, $\mathcal{T} =< ao_1, ao_2, ..., ao_n >$.

2.2 Recommendation Algorithm

The proposed recommender is based on previous work [16], in which the past activities performed by a user were modelled as a timeline, $\mathcal{T}$, and the objective was to recommend the next activity to a user to perform. Here, we extend this approach to recommend the next *sequence of activities* for users to perform, $\mathcal{T}_{rec} =< ao_{rec_1}, ao_{rec_2}, ..., ao_{rec_L} >$.

Referring to Algorithm 1, a sequence of activities at a given recommendation time (RT) are generated as follows. The most recent activity object performed by the user, referred to as the *current activity object*, ao_c, is initialised as the activity object occurring at time RT in the user's timeline. The *current timeline*, $\mathcal{T}_c$, is then extracted from the user's timeline; it consists of the subsequence of the N activity objects occurring prior to ao_c and ends with ao_c (Step 1).

The recommendation of each activity object ao_{rec_i} in $\mathcal{T}_{rec}$ is performed iteratively (Step 4) as follows (see [16] for details). For each previous occurrence in the user's timeline of an activity with the same name as ao_c (e.g. 'Italian Food'), a *candidate timeline* ($\mathcal{T}_j$) is extracted (Step 5). Let $\mathscr{T}$ be the set of all candidate timelines in a given iteration. A two-level edit distance ($d(.,.)$) between each candidate and the current timeline is computed [15]; based on these distances, a score (Eqn. 1) is assigned to the activity that occurs immediately after each candidate timeline $\mathcal{T}_j$ in $\mathscr{T}$ (Steps 7–8).

Algorithm 1: ***SeqNCSeqRec***

Input: User, u; user's past timeline, $\mathcal{T}$; recommendation time, RT; current activity object, ao_c; N-count value, N
Output: a recommended timeline (sequence) $\mathcal{T}_{rec}$ of L activity objects, $\mathcal{T}_{rec} =< ao_{rec_1}, ao_{rec_2}, ...ao_{rec_i}..., ao_{rec_L} >$

1. Extract the current timeline $\mathcal{T}_c$ from $\mathcal{T}$; the final element of $\mathcal{T}_c$ is ao_c
2. $\mathcal{T}_{rec} \leftarrow <>$
3. $i \leftarrow 1$
4. **while** $i \leq L$ **do**
5. Extract candidate timelines $\mathscr{T}$ from $\mathcal{T}$ (each $\mathcal{T}_j \in \mathscr{T}$ ends with an activity object ao_f^j such that ao_f^j.name = ao_c.name)
6. $\mathcal{R} \leftarrow \{\}$
7. **for** each $\mathcal{T}_j \in \mathscr{T}$ **do** $\mathcal{R} \leftarrow \mathcal{R} \cup ao_{f+1}^j$
8. **for** each $ao \in \mathcal{R}$ **do** Compute $Score(ao)$
9. ao_{rec_i}.name $\leftarrow$ top-1(ao.name : $ao \in \mathcal{R}$)
10. Compute and assign features to ao_{rec_i}
11. $\mathcal{T}_{rec} \leftarrow append(\mathcal{T}_{rec}, ao_{rec_i})$
12. $\mathcal{T}_c \leftarrow append(\mathcal{T}_c, ao_{rec_i})$
13. $RT \leftarrow ao_{rec_i}$.time
14. $i \leftarrow i + 1$
15. **return** $\mathcal{T}_{rec}$

From this set of scored activity objects, the top-1 activity name with the highest score is returned as the name for ao_{rec_i} in $\mathcal{T}_{rec}$ (Step 9). The values for the other features of ao_{rec_i} are then computed (Step 10) based on the average values for each feature from the user's past timeline. For example, if the recommended activity name is eating 'Italian Food', the time at which this activity should occur (ao_{rec_i}.time) is calculated as follows. The median difference between all occurrences of 'Italian Food' and the immediately preceeding activity in the user's past timeline is calculated; ao_{rec_i}.time is then given by the current recommendation time (RT) plus this difference.

Before the next iteration of the algorithm, ao_{rec_i} is appended to the current timeline $\mathcal{T}_c$ (and becomes the current activity object in the next iteration) (Step 12) and the recommendation time (RT) is set to ao_{rec_i}.time (Step 13). Thus, the L activity objects in the recommended timeline $\mathcal{T}_{rec}$ are generated in L iterations.

$$Score(ao) = 1 - \frac{d(\mathcal{T}_j, \mathcal{T}_c) - \min\limits_{\mathcal{T}_p \in \mathscr{T}} d(\mathcal{T}_p, \mathcal{T}_c)}{\max\limits_{\mathcal{T}_p \in \mathscr{T}} d(\mathcal{T}_p, \mathcal{T}_c) - \min\limits_{\mathcal{T}_p \in \mathscr{T}} d(\mathcal{T}_p, \mathcal{T}_c)} . \quad (1)$$

2.2.1 Distance between Timelines. For the purpose of determining the similarity between two timelines $\mathcal{T}_1$ and $\mathcal{T}_2$, the two-level similarity algorithm proposed in our earlier work [15] is used. This algorithm first computes the minimum cost of rearranging the activities to achieve the same activity sequence and then aligns the values of the features of the corresponding activity objects. See [15] for further details on this approach.

2.2.2 N-count matching. The matching unit determines the length of the subsequences to be considered when calculating the distances between timelines. The *SeqNCSeqRec* algorithm uses the N-count matching approach as proposed in [16]. Thus, the N activity objects in the timeline preceding the current activity object form the current timeline (and likewise for candidate timelines). Note that the optimal value of N for each user will differ, depending on the degree of repetition and regularity of activities performed by each.

3 EVALUATION

We first describe the dataset used to construct activity timelines for users and the experimental methodology employed. This is followed by an evaluation of the proposed N-count based sequence recommender.

3.1 Dataset

For our experiments, we used a subset of the Gowalla checkins dataset [21]. The complete dataset obtained contains around 36 million checkins, 2.8 million locations and 0.3 million users. Every checkin is bound to a specific location and timestamp. A subset of these locations have categories assigned to them, such as, 'Italian Food', 'Bookstore', 'City Park', etc. These locations also have contextual features such as latitude, longitude, number of users checking in to it, number of photos taken at the location, etc. In relation to our recommendation framework, each of the location categories is considered as an 'activity name' and the recommendations made will be sequences of these categories. Hence, for evaluation, we select only those checkins locations which have assigned categories.

Further, categories are organised in a three-level hierarchy, consisting of 7, 134 and 151 level 1, 2, and 3 categories, respectively. For example, the level 1 category 'Food' has child categories 'African', 'American', 'Asian', 'Coffee Shop', etc. at level 2, while 'Coffee Shop' has child categories 'Starbucks' and 'Dunkin Donuts' at level 3. Given our objective is to recommend activities (categories) to users, we consider level 2 categories as the most suitable level of granularity, and hence any checkin locations with level 3 categories are assigned the parent category at level 2. As such, the names of activity objects in user timelines are given by the level 2 categories of the locations checked in to by users.

Since the characteristics of the timelines on weekdays and weekends are different, here we considered data corresponding to weekdays only. To address multiple consecutive checkins by users at the same location, we merged such checkins for a given user if they had the same category, were less than 600 meters apart and occurred within an interval of 10 minutes. Further, we selected only those users which have checkin data for at least 50 days with a minimum of 10 checkins per day. The sampled dataset had 916 users with 2.7 million checkins in total. The median number of checkins per day for users varied from 11–134, while the median number of distinct categories of checkins per day for users varied from 4–58.

3.2 Methodology

An offline evaluation was conducted for the proposed recommendation approach. Each user's complete timeline was split into training and test timelines, where the test timeline contained data for the most recent 20% of available days. For each user, a recommended

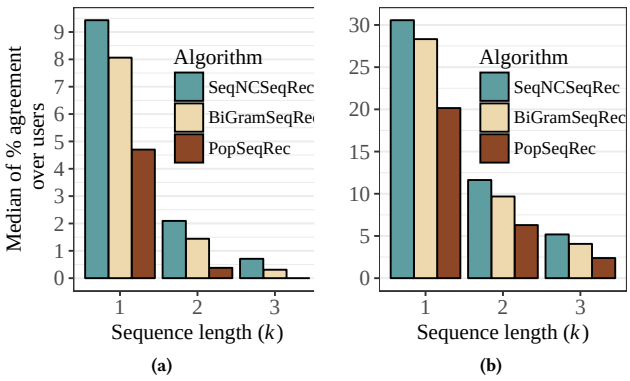


Figure 1: Median percentage agreements for recommended sequences for *SeqNCSeqRec* and baseline algorithms using timelines constructed from categories at (a) level 2 and (b) level 1 in the hierarchy.

sequence of categories of length 3 was generated at different *recommendation times* (*RT*s), which corresponded to the end time of each activity object in the test timeline. Recommendation performance is evaluated using *agreement @ k* ($k = 1, 2, 3$) which is the percentage of *RT*s for a user where the first k categories in the recommended sequence and the actual sequence are an exact match.

For the computation of two-level edit distances between timelines, the following operation costs and feature weights were used: $c_{ins} = c_{del} = 1$, and $c_{sub} = 2$; $w_{category} = 2$, $w_{start-time} = 1$, $w_{popularity} = 1$, $w_{location} = 1$. These weights were set according to their hypothesised importance from the perspective of comparing timelines; for example, the weight associated with updating the category was set to the highest value since this is clearly a key consideration when computing distances between timelines. See [15] for details on the two-level edit distance approach.

3.3 Recommendation Performance

The performance of our proposed sequence-based N-count sequence recommendation algorithm (*SeqNCSeqRec*) is compared to the following baselines:

- The bi-gram-based sequence recommender (*BiGramSeqRec*) is based on the Markov assumption that the next activity depends only on the current activity. For each user, the frequency of occurrence of all activity name (category) bi-grams in the user's timeline are computed. For a given *RT*, a sequence of activity objects is recommended iteratively as per *SeqNCSeqRec* except that, at each iteration, the most frequently occurring bi-gram beginning with the current activity name is identified, and the recommended activity is simply that of the second element of this bi-gram. Such Markov-based approaches have proved to be quite successful in modelling sequences in previous studies [9].
- For a given user and *RT*, at each iteration of the algorithm, the popularity-based sequence approach (*PopSeqRec*) recommends the activity that the user performed most frequently at that time in the past.

3.3.1 Algorithm Performance. Figure 1(a) shows the median percentage agreements (k = 1, 2, 3) over all users for the proposed *SeqNCSeqRec* recommender and the two baselines. For *SeqNCSeqRec*, the results shown correspond to the optimal value of N-count for each user. It is clear from these results that the proposed approach significantly outperforms the baseline approaches. For example, *SeqNCSeqRec* improves upon *BiGramSeqRec* by 16.98%, 45.38%, and 129.3% for recommended sequences of length 1, 2, and 3, respectively, and improves upon *PopSeqRec* by more than 100% in all cases. Differences in results between the proposed and baselines algorithms are statistically significant (Wilcoxon-Mann-Whitney rank sum test) at the $p<.05$ level. The results also indicate that performance declines when larger sequences are recommended, which is to be expected, given the increased challenges involved in making such recommendations.

While the above findings are promising, it can be seen that the percentage agreements achieved by all algorithms are relatively low; for example, the percentage agreement is 9.5% for sequence lengths of 1 using *SeqNCountSeqRec*. We make the following observations in this regard. Firstly, as described in the previous section, in order to generate a sequence of recommendations, only the top-1 recommended activity is considered at each iteration of the *SeqNCSeqRec* algorithm. In addition, the evaluation is based on only a single recommended sequence being made to users, which clearly represents a strict approach.

Secondly, while many (although not all) level 2 categories are semantically similar, they are not considered a match according to the evaluation metric. For example, consider the level 2 categories 'Mexican' and 'South American/Latin' which relate to dining and are children of the level 1 category 'Food'. From a recommendation perspective, these different types of dining experiences are clearly related and (arguably) should represent a match. Thus, we also evaluate our recommender when all checkin locations are mapped to level 1 categories in the hierarchy – i.e. user timelines are constructed from activity objects with names given by the level 1 categories of locations checked in to by users. The results are shown in Figure 1(b). While similar trends as before are seen, the percentage agreements achieved are much greater; for example, over 30% for *SeqNCSeqRec* compared to the previous 9.5% for sequences of length 1. The 'true' performance of the recommender lies somewhere in between these values (since not all level 2 categories are semantically related); a further analysis of this matter is left to future work.

3.3.2 Performance across Users. A key intuition behind our approach is that the next activities performed by individual users depends, to a lesser or greater extent, on their past activity patterns. In the proposed *SeqNCSeqRec* recommendation algorithm, the number of past activities to be considered when generating recommendations is determined by the N-count value (see Section 2.2). In previous work [16], where the task was to recommend a single activity to users, it was seen that the optimal N-count value varied across users. In this section, we investigate whether a similar affect is seen when recommending sequences of activities to users.

As per [16], we hypothesise three distinct groups of users to capture the degree to which past activity patterns reflect future activity performance – Group 1: next activities are based on the current activity only (N-count = 0); Group 2: next activities are based on the current activity and a small number of past activities (N-count lies in the interval [1,4]); and Group 3: next activities are based on the current activity and a larger number of past activities (N-count = 5+).

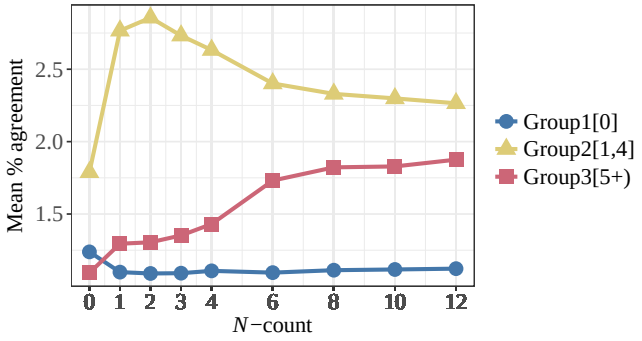


Figure 2: Mean percentage agreement for recommended sequences over users in each group.

In this experiment, users were assigned to one of the above groups based on the range in which their optimal N-count value appears (optimal in the sense that best percentage agreement was seen for sequences of length 3). Overall, 421, 374 and 121 users were assigned to Groups 1, 2 and 3, respectively. Results are shown in Figure 2. It can be seen that the mean recommendation performance for Group 1 users (46% of all users) was significantly lower than that seen for users in the other groups. This finding is to be expected, since it indicates that it is easier to recommend sequences of activities to users which are more consistent in their activity patterns. Thus, it can be concluded that adopting a personalised approach for users, by selecting the optimal N-count value for each user, is important. While it is not feasible to determine this value by experiment for large user bases, an approach to automatically learn a suitable value for individual users such as proposed in previous work [16] can be applied.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we have expanded on our previous work to suggest sequences of activities for users based on past activity patterns. Notwithstanding the strict evaluation metric used in this work, the proposed approach shows promising performance and outperforms the baseline algorithms considered. In future work, we will investigate collaborative approaches in which candidate timelines will be drawn from the activities of other users in the system. Further, we will consider new approaches to suggest sequences of activities (for example, using RNNs) and investigate the recommendation of context (for example, where, when, with whom etc.) associated with each of the suggested sequence of activities.

5 ACKNOWLEDGMENTS

The Insight Centre for Data Analytics is supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289.

REFERENCES

[1] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alex Tuzhilin. 2011. Context-Aware Recommender Systems. *AI Magazine* 32, 3 (2011).

[2] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in Location-based Social Networks: A Survey. *Geoinformatica* 19, 3 (July 2015), 525–565. https://doi.org/10.1007/s10707-014-0220-8

[3] Thorsten Bohnenberger and Anthony Jameson. 2001. When Policies Are Better Than Plans: Decision-theoretic Planning of Recommendation Sequences. In *Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI '01)*. ACM, New York, NY, USA, 21–24. https://doi.org/10.1145/359784.359829

[4] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *Comput. Surveys* 47, 2, Article 26 (Nov. 2014), 35 pages. https://doi.org/10.1145/2652481

[5] Joan Borràs, Antonio Moreno, and Aida Valls. 2014. Intelligent Tourism Recommender Systems: A survey. *Expert Systems with Applications* 41, 16 (2014), 7370 – 7389. https://doi.org/10.1016/j.eswa.2014.06.007

[6] Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Thomas Schievenin. 2013. *Context-Aware Points of Interest Suggestion with Dynamic Weather Data Management*. Springer-Verlag, Cham, 87–100. https://doi.org/10.1007/978-3-319-03973-2_7

[7] Dimitrios Buhalis. 2003. *eTourism: Information technology for strategic tourism management*. Pearson Education.

[8] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 714–722. https://doi.org/10.1145/2339530.2339643

[9] Mukund Deshpande and George Karypis. 2004. Selective Markov Models for Predicting Web Page Accesses. *ACM Transactions on Internet Technology* 4, 2 (May 2004), 163–184. https://doi.org/10.1145/990301.990304

[10] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. Mobile recommender systems in tourism. *Journal of Network and Computer Applications* 39 (2014), 319 – 333. https://doi.org/10.1016/j.jnca.2013.04.006

[11] Ramanathan Guha, Vineet Gupta, Vivek Raghunathan, and Ramakrishnan Srikant. 2015. User Modeling for a Personal Assistant. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, New York, NY, USA, 275–284. https://doi.org/10.1145/2684822.2685309

[12] Cathal Gurrin, Alan F. Smeaton, and Aiden R. Doherty. 2014. LifeLogging: Personal Big Data. *Foundations and Trends in Information Retrieva* 8, 1 (June 2014), 1–125. https://doi.org/10.1561/1500000033

[13] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22, 1 (Jan. 2004), 5–53. https://doi.org/10.1145/963770.963772

[14] Katerina Kabassi. 2010. Personalizing Recommendations for Tourists. *Telematics and Informatics* 27, 1 (2010), 51 – 66. https://doi.org/10.1016/j.tele.2009.05.003

[15] Gunjan Kumar, Houssem Jerbi, Cathal Gurrin, and Michael P. O'Mahony. 2014. Towards Activity Recommendation from Lifelogs. In *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services (iiWAS '14)*. ACM, New York, NY, USA, 87–96. https://doi.org/10.1145/2684200.2684298

[16] Gunjan Kumar, Houssem Jerbi, and Michael P. O'Mahony. 2016. Personalised Recommendations for Modes of Transport: A Sequence-based Approach. *The 5th ACM SIGKDD International Workshop on Urban Computing (UrbComp 2016)* (2016).

[17] Carlos Lamsfus, David Martin, Zigor Salvador, Alex Usandizaga, and Aurkene Alzua-Sorzabal. 2009. Human-Centric Ontology-Based Context Modelling In Tourism. *Mediterranean Conference on Information Systems* (2009).

[18] Carlos Lamsfus, Dan Wang, Aurkene Alzua-Sorzabal, and Zheng Xiang. 2015. Going Mobile. *Journal of Travel Research* 54, 6 (2015), 691–701. https://doi.org/10.1177/0047287514538839

[19] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining User Similarity Based on Location History. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '08)*. ACM, New York, NY, USA, Article 34, 10 pages. https://doi.org/10.1145/1463434.1463477

[20] H. Liu, L. Y. Wei, Y. Zheng, M. Schneider, and W. C. Peng. 2011. Route Discovery from Mining Uncertain Trajectories. In *11th IEEE International Conference on Data Mining Workshops*. 1239–1242. https://doi.org/10.1109/ICDMW.2011.149

[21] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. 2013. Personalized Point-of-interest Recommendation by Mining Users' Preference Transition. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 733–738. https://doi.org/10.1145/2505515.2505639

[22] Brian McFee and Gert RG Lanckriet. 2011. The Natural Language of Playlists. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*.

[23] Brian McFee and Gert RG Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*.

[24] Antonio Moreno, Aida Valls, David Isern, Lucas Marin, and Joan Borràs. 2013. SigTur/E-Destination: Ontology-based Personalized Recommendation of Tourism and Leisure Activities. *Engineering Applications of Artificial Intelligence* 26, 1 (Jan. 2013), 633–651. https://doi.org/10.1016/j.engappai.2012.02.014

[25] James Pitkow and Peter Pirolli. 1999. Mining Longest Repeating Subsequences to Predict World Wide Web Surfing. In *Proceedings of the 2nd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 2 (USITS'99)*. USENIX Association, Berkeley, CA, USA, 13–13. http://dl.acm.org/citation.cfm?id=1251480.1251493

[26] Francesco Ricci. 2002. Travel Recommender Systems. *IEEE Intelligent Systems* (2002).

[27] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. 2010. *Recommender Systems Handbook* (1st ed.). Springer-Verlag New York, Inc., New York, NY, USA.

[28] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *Journal of Machine Learning Research* 6 (Dec. 2005), 1265–1295. http://dl.acm.org/citation.cfm?id=1046920.1088715

[29] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of Predictability in Human Mobility. *Science* 327, 5968 (2010), 1018–1021. https://doi.org/10.1126/science.1177170

[30] Chih-Hua Tai, De-Nian Yang, Lung-Tsai Lin, and Ming-Syan Chen. 2008. Recommending Personalized Scenic Itinerary with Geo-tagged Photos. In *IEEE International Conference on Multimedia and Expo*. 1209–1212. https://doi.org/10.1109/ICME.2008.4607658

[31] Iis P. Tussyadiah and Dan Wang. 2016. Tourists' Attitudes toward Proactive Smartphone Systems. *Journal of Travel Research* 55, 4 (2016), 493–508. https://doi.org/10.1177/0047287514563168

[32] Artem Umanets, Artur Ferreira, and Nuno Leite. 2014. GuideMe – A Tourist Guide with a Recommender System and Social Interaction. *Procedia Technology* 17 (2014), 407 – 414. https://doi.org/10.1016/j.protcy.2014.10.248 Conference on Electronics, Telecommunications and Computers – CETC 2013.

[33] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2010. Smart Itinerary Recommendation Based on User-generated GPS Trajectories. In *Proceedings of the 7th International Conference on Ubiquitous Intelligence and Computing (UIC'10)*. Springer-Verlag, Berlin, Heidelberg, 19–34. http://dl.acm.org/citation.cfm?id=1929661.1929669

[34] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2012. Social Itinerary Recommendation from User-generated Digital Trails. *Personal Ubiquitous Comput.* 16, 5 (June 2012), 469–484. https://doi.org/10.1007/s00779-011-0419-8

[35] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09)*. ACM, New York, NY, USA, 791–800. https://doi.org/10.1145/1526709.1526816

Itinerary Recommendation for Cruises: User Study

Diana Nurbakova*
LIRIS - INSA Lyon
20 avenue Albert Einstein
Villeurbanne 69621 cedex, France
diana.nurbakova@insa-lyon.fr

Léa Laporte
LIRIS - INSA Lyon
20 avenue Albert Einstein
Villeurbanne 69621 cedex, France
lea.laporte@insa-lyon.fr

Sylvie Calabretto
LIRIS - INSA Lyon
20 avenue Albert Einstein
Villeurbanne 69621 cedex, France
sylvie.calabretto@insa-lyon.fr

Jérôme Gensel
Université Grenoble Alpes, CNRS, Grenoble INP, LIG
Grenoble F-38000, France
jerome.gensel@univ-grenoble-alpes.fr

ABSTRACT

Vacations and leisure activities constitute an important part of human life. Nowadays, a lot of attention is paid to cruising, that is reported to be a favourite vacation choice for families with kids and for Millenials. Like other distributed events (events that gather multiple activities distributed in space and time under one umbrella) such as big festivals, conventions, conferences etc., cruises offer a vast variety of simultaneous on-board activities for all ages and tastes. This results in a cruiser's information overload, in particular given a very limited availability of activities. Recommender systems appear as a desirable solution in such an environment. Due to the number of time constraints, it is more convenient to get a personalised itinerary of activities rather than a list of top-*n*. In this paper, we present a user study conducted in order to create a preliminary dataset that simulates users' attendance of a cruise and sheds the light on the activity selection behaviour. We discuss challenges faced by the itinerary recommendation and illustrate them with user study examples.

CCS CONCEPTS

• **Information systems** → **Personalization**;

KEYWORDS

recommendation of leisure activities, itinerary recommendation

1 INTRODUCTION

Nowadays, the field of leisure activities experiences a substantial growth. In this context, a rising phenomenon we are witnessing is distributed events that gather various activities under one umbrella. They attract more and more attendees. Examples of such events are cruises, festivals, big conferences, conventions, etc.

Attendees of distributed events are overwhelmed with the number of ongoing parallel activities and are looking for personalised experience. Recommender systems appear as a natural solution in such an environment. It is to note that given the density of activities and their limited availability, participants are interested in a personalised itinerary (a sequence of activities to undertake) rather than in a list of top-*n* activities that may compete in terms of time.

*D. Nurbakova held a doctoral fellowship from la Région Auvergne-Rhône-Alpes.

In this work, we consider a case of a cruise. According to Florida-Caribbean Cruise Association (F-CCA) [6], about 25.3M passengers are expected to cruise globally in 2017, showing a 7% average annual passenger growth rate over the last 30 years. Cruising has become a preferred vacation choice for families, especially with kids, making cruisers population younger and more diverse than non-cruisers. F-CCA reports [6] that cruising is the favourite choice of Millennials and Generation X. Cruisers appreciate the opportunity to relax and get away from it all, see and do new things. Cruise lines offer a vast variety of on-board activities, as well as in ports of call.

In this paper, we focus on the itinerary recommendation and present a user study based on a 7-night Disney Fantasy cruise. More precisely, we aim at answering the following research questions.

RQ1: What is itinerary recommendation and what makes it challenging?

RQ2: What are the characteristics of the data treated by itinerary recommendation? Is there any dataset that could be used as is?

The remainder of the paper is organised as follows. In Section 2 we define the itinerary recommendation problem and the challenges it faces. Section 3 gives an overview of existing datasets, presents our user study that simulates users' attendance of a cruise and discussion over conducted analysis. Section 4 concludes the paper.

2 PROBLEM STATEMENT AND CHALLENGES

In this paper, we aim at finding a personalised itinerary for a given user that maximises his satisfaction and takes into account spatio-temporal constraints. More precisely, given a set of activities with their locations, descriptions, time windows of their availability, duration, and a vector of categories, a set of users, and users' history (attendance) binary matrix, find a feasible sequence of activities (or itinerary) that maximises the user's satisfaction for every given user. User's satisfaction with respect to an itinerary is defined as the sum of the user's satisfaction scores regarding all the activities within the itinerary. For more details on the itinerary recommendation problem, see [9].

Itinerary recommendation faces the following **challenges**.

C-1: Implicit Feedback. Given that activities are happening in future as in the case of event recommendation [8], there is very little information to handle and there is much less user-item interactions than in traditional recommendation scenarios. We deal with implicit feedback, implying that the degree to which a user likes or not an

item is not known. The use of multiple contexts may increase the recommendation performance of the algorithms.

C-2: Interest vs. Attendance. Due to the limited availability and multiple parallel activities, we deal with attendance bias, as a user may miss an activity of his/her interest or in contrast, may join an activity that does not represent a particular interest to him/her.

C-3: List vs. Itinerary. Activities are competitive and short-lived, which results in the user's preference for one activity over the others in a given time slot. In this context, an itinerary (a feasible sequence of activities) is more desirable than a list of interesting activities.

We will illustrate the challenges in the next section.

3 USER STUDY

In this section, we formulate a list of characteristics of a dataset satisfying the needs of the target problem, provide a comparison of available datasets (see Tab. 1) and describe a user study conducted in order to collect data with desirable characteristics.

3.1 Data Characteristics and Existing Datasets

We categorise the existing datasets w.r.t. the focus of data into 3 groups: *Single Item*, *Schedule*, and *Sequence*. We define a list of characteristics (column "Characteristics" in Tab. 1) based on the activity attributes and consecutive nature of performed activities during distributed events. We cluster the characteristics into 5 types w.r.t. the entity they describe: item (unit under consideration), sequence (ordered sequence of items), user (information about users), user-item (user-item interactions), and user-user (relations between users). We distinguish 5 essential characteristics (given in italics in Tab. 1): (1) time windows (start and end time of activity availability), (2) coordinates (geographical location of an activity), (3) service time (duration of an activity), (4) categories (associated categories), (5) users historical data. Though we indicate only 5 elements as essentials, all the others listed in Tab. 1 are also important as they may enhance the recommendation. As it can be seen, none of the existing datasets contains all the essential characteristics. Thus, we have made an attempt to create an integral dataset that contains all the required features and provides an insight into users' behaviour.

3.2 Data Collection

In order to collect required data, we have performed a user study via online survey. Participants were recruited via a link to the online questionnaire sent by email to several research and university mailing lists. The claimed aim of the study was to create a dataset that simulates cruise attendance and could be used in order to make personalised recommendations of itineraries. The list of activities used in the survey was taken from the personal navigators of Disney's Fantasy 7-nights Eastern Caribbean cruise. Activities dedicated exclusively for kids have been excluded from the current list of activities. The original personal navigators can be found online[3]. The deck plan of the ship can be found on the web[4]. The

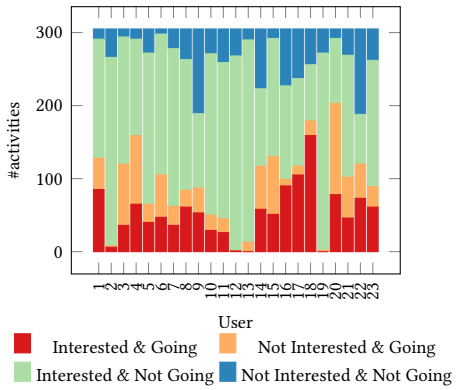


Figure 1: Distribution of interest in activities and attendance per user.

questionnaire consisted of 4 parts. The overview of the survey with examples of questions is given in Tab. 2.

Thus, 23 contributions were collected. Statistics concerning the participants are provided in Tab. 3. The main statistics of the dataset are given in Tab. 4. The average duration of an activity is 45 minutes. The average number of ongoing simultaneous activities is 5.

3.3 Data Analysis

The conducted user study gives a more practical insight into personalised itinerary recommendation and the activity selection process. In the following, we illustrate the challenges from Section 2.

C-2: Interest vs. Attendance. Figure 1 displays the user-wise distribution of the number of activities a user: (1) was interested in ($rating \geq 4$ or $rating = 3$ if the highest rating given by the user to any activity is equal to 3) and joined (*Interested & Going*), (2) was interested in but did not join (*Interested & Not Going*), (3) was not interested in but joined (*Not Interested & Going*), and (4) was not interested in and did not join (*Not Interested & Not Going*)[5]. The chart shows evidence that individuals miss many activities that represent interest to them. Thus, the number of *Interested & Not Going* activities is almost twice higher (1.7621) than *Interested & Going*. It is also surprising that *Not Interested & Going* activities constitute about 43% of all joined activities.

C-3: List vs. Itinerary. Let us consider the following settings. We compare several top-n item recommendation algorithms against itinerary recommendation from the literature. As history data we consider a binary attendance matrix.

- *Category-based*: This algorithm ranks the candidate activities based on their weighted frequency of corresponding categories.

- *Content-based*: The candidate activities are ranked in descendant order of their textual similarity with the user's past activities. An activity is represented as a TF-IDF vector. The user's profile is built over TF-IDF vectors of activities joined by the user in the past.

- *Logistic Regression*: We fed a vector of aforementioned scores into a logistic regression model.

[1] Yelp challenge dataset, http://www.yelp.com/dataset_challenge
[2] https://github.com/jalbertbowden/foursquare-user-dataset
[3] http://disneycruiselineblog.com/2015/07/personal-navigators-7-night-eastern-caribbean-cruise-on-disney-fantasy-itinerary-a-june-20-2015/
[4] http://disneycruiselineblog.com/ships/deck-plans-disney-dream-disney-fantasy/

[5] Ratings are used only for this part of the study. We do not consider them in estimation of user's interest in activities, as we assume there exist only binary attendance matrix.

Table 1: Comparison of the available datasets.

		Single Item									Schedule			Sequence	
Entity	Characteristic	TREC CS'13 [3]	TREC CS'14 [4]	TREC CS'15 [2]	Yelp [1]	Foursquare_1 [14]	Foursquare_2 [2]	Flickr [11]	Twitter [5]	Meetup [7]	MCTOPMTW [10]	Other OP-TW [12]	Other OP-based [12]	TripBuilder [1]	GeoLife [15]
Item	*Time windows*				✓			✓			✓	✓			
	Coordinates	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	Service Time										✓	✓	✓		
	Categories									✓	✓			✓	
	Price					✓					✓			✓	
	Item Additional Attributes				✓			✓		✓					
	Description							✓	✓						
Sequence	Time budget										✓	✓	✓		
	Starting/Ending Point										✓	✓	✓		✓
	Tour Additional Attributes			✓											✓
User	User's personal data			✓						✓					
User-Item	*Historical Data*	✓	✓	✓		✓	✓	✓	✓	✓				✓	✓
	Score	✓	✓	✓		✓	✓				✓	✓	✓		
User-User	Social links				✓	✓	✓			✓					

Table 2: Description of the parts of the survey. Qnt denotes the number of questions in a section.

Section	Qnt	Description	Question Examples
User Profile	10	Questions on basic user's features and their cruising experience	Your gender: □Female ☑Male Have you already experienced DCL (Disney Cruise Line)? Are you aiming to attend the maximum amount of activities mentioned in your Personal Navigator or just a few must-see?
Users Preferences	311	User's evaluation of a list of proposed activities by selecting one of the grades for the listed activities: 1 - Never (not interested at all and won't recommend to anyone to attend it); 2 - Not interested; 3 - Neutral; 4 - Interested; 5 - Won't miss	**Sailing Away. Don't Miss Event.** *Description:* It's time to go Sailing Away! Join Mickey and Minnie along with Tinker Bell and the rest of the gang as they welcome you abroad the Disney Fantasy. *Available:* Day 1, 16:30-17:15, *Location*: Deck Stage *Never* ○○○○● *Won't miss*
Itinerary Planner	593	Organisation of the activities into a day-wise itinerary. Given an ordered list of activities with their availability hours, the respondents were asked to indicate their intention to join the activity or not by clicking on "Going" or "Not going".	Event: 11:30 - 15:00. Character Meet & Greet Ticket Distribution. Category: Characters. Location: Port Adventures Desk. Don't Miss Event — Going: ☑ — Not going: □
Afterwards	5	Conclusion questions	When you were having a choice among different activities of your interest, did you consider the distance to the venue while making your choice? How do you usually manage the list of activities to perform during your vacations?

- *ILS+Scores*: We also tested a state-of-the-art itinerary construction algorithm [9] that is based on the Iterated Local Search (ILS) algorithm [13] with activities scores calculated using hybrid scores (content-based, category-based and time-based) and transition probabilities between activities.

Table 3: Participants Statistics

Statistic	Value
# Female users	7
# Users already experienced DCL	1
# Users already experienced any cruise	4
# Users considering the distance between venues	8
Managing Activities. Not-to-miss List : Daily planning : No planning	14 : 4 : 5
Age group: 21-30 : > 30	16 : 7

Table 4: Dataset Statistics

# Activities	# Days	# Users	# Locations	# Categories
593	7	23	47	52

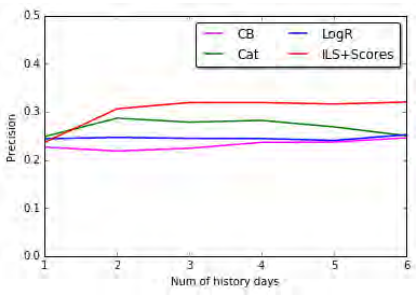


Figure 2: Precision w.r.t. the number of history days.

The algorithms were evaluated in terms of their precision. We returned top-20 activities for each day[6] using top-*n* recommendation algorithms. Figure 2 displays the recommendation power of each algorithm with varying number of history days (from 1 to 6). Itinerary recommendation algorithm shows higher precision, proving that an itinerary satisfies better the user's needs.

4 DISCUSSION AND CONCLUSION

In this paper we have considered the problem of personalised itinerary recommendation with special interest for cruises. We have distinguished the characteristics of data used in itinerary recommendation and have presented an overview of available datasets. To the best of our knowledge, this is the first attempt to classify and summarise the existing datasets, and describe them with respect to the aforementioned characteristics. Moreover, we have undertaken a user study in order to build a preliminary dataset that satisfies all the characteristics and that helps to understand individuals' behaviour in activity selection process. Though the discussed dataset is not large-scale, the undertaken user study reveals general trends of users' behaviour while on board of a cruise or while attending a distributed event. Moreover, we have discussed the challenges faced by the problem of itinerary recommendation and have illustrated them with the performed data analysis.

As future work, we plan to create a dataset via crowdsourcing using CrowdFlower platform. The characteristics presented in Sec. 3.1 will serve as the basis for the new dataset. Another direction of future work consists in proposing more accurate solution for the itinerary recommendation that would embrace all the sides and address all the challenges of the itinerary recommendation.

[6]The average number of joined activities per day is 18.

REFERENCES

[1] Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2013. Where Shall We Go Today?: Planning Touristic Tours with Tripbuilder. In *Proc. of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. 757–762.

[2] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen Voorhees. 2015. Overview of the TREC 2015 Contextual Suggestion Track. In *Proc. of the 24th Text REtrieval Conference (TREC 2015)*.

[3] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Paul Thomas, Nicole Simone, and Ellen Voorhees. Overview of the TREC 2013 Contextual Suggestion Track. In *NIST Special Publication 500-302: The Twenty-Second Text REtrieval Conference Proceedings (TREC 2013)* (2013), Ellen M. Voorhees (Ed.).

[4] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Paul Thomas, and Ellen Voorhees. Overview of the TREC 2014 Contextual Suggestion Track. In *NIST Special Publication 500-308: The Twenty-Third Text REtrieval Conference Proceedings (TREC 2014)* (2014), Ellen M. Voorhees and Angela Ellis (Eds.).

[5] Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1277–1287.

[6] Florida-Caribbean Cruise Association (FCCA). 2017. Cruise Industry Overview. 11200 Pines Blvd., Suite 201 - Pembroke Pines, Florida 33026. http://www.f-cca.com/downloads/2017-Cruise-Industry-Overview-Cruise-Line-Statistics.pdf

[7] Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. 2012. Event-based Social Networks: Linking the Online and Offline Social Worlds. In *Proc. of the 18th ACM SIGKDD conference on Knowledge Discovery and Data Mining* (2012) *(KDD'12)*.

[8] Augusto Q. Macedo, Leandro B. Marinho, and Rodrygo L.T. Santos. 2015. Context-Aware Event Recommendation in Event-based Social Networks. In *Proc. of the 9th ACM Conference on Recommender Systems (RecSys '15)*. 123–130.

[9] Diana Nurbakova, Léa Laporte, Sylvie Calabretto, and Jérôme Gensel. 2017. Recommendation of Short-Term Activity Sequences During Distributed Events. *Procedia Computer Science* 108 (2017), 2069 – 2078. International Conference on Computational Science, {ICCS} 2017, 12-14 June 2017, Zurich, Switzerland.

[10] Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. 2013. The Multiconstraint Team Orienteering Problem with Multiple Time Windows. *Transportation Science* 47, 1 (Feb. 2013), 53–63.

[11] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. YFCC100M: The New Data in Multimedia Research. *Commun. ACM* 59, 2 (2016), 64–73.

[12] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *Eur J Oper Res* 209, 1 (2011), 1 – 10.

[13] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. 2009. Iterated Local Search for the Team Orienteering Problem with Time Windows. *Comput. Oper. Res.* 36, 12 (Dec. 2009), 3281–3290.

[14] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory Cultural Mapping Based on Collective Behavior Data in Location-Based Social Networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 30.

[15] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proc. of the 18th International Conference on World Wide Web (WWW '09)*. 791–800.

Co-Staying: A Social Network for Increasing the Trustworthiness of Hotel Recommendations

Catalin-Mihai Barbu
University of Duisburg-Essen
Duisburg, Germany
catalin.barbu@uni-due.de

Jürgen Ziegler
University of Duisburg-Essen
Duisburg, Germany
juergen.ziegler@uni-due.de

ABSTRACT
Recommender systems attempt to match users' preferences with items. To achieve this, they typically store and process a large amount of user profiles, item attributes, as well as an ever-increasing volume of user-generated feedback about those items. By mining user-generated data, such as reviews, a complex network consisting of users, items, and item properties can be created. Exploiting this network could allow a recommender system to identify, with greater accuracy, items that users are likely to find attractive based on the attributes mentioned in their past reviews as well as in those left by similar users. At the same time, allowing users to visualize and explore the network could lead to novel ways of interacting with recommender systems and might play a role in increasing the trustworthiness of recommendations. We report on a conceptual model for a multimode network for hotel recommendations and discuss potential interactive mechanisms that might be employed for visualizing it.

Author Keywords
Recommender systems; tourism; personalization; trust; multimode networks; trustworthiness

ACM Classification Keywords
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*evaluation/methodology, graphical user interfaces (GUI), user-centered.*

INTRODUCTION
Recommender systems (RS) typically build and maintain network representations of the data they store about their users and product catalogs. One of the earliest and most well-known examples is the "item-to-item" model introduced by Amazon in its RS, which enabled the company to show, using collaborative filtering techniques, what *other* products were purchased by users who bought a certain product [8]. Linking user preferences with items and subsequently modeling the various relationships that arise between them can increase the accuracy of recommendations. Most commonly, RS employ 1-mode networks, meaning that all nodes are of the same type (e.g., users). Less frequent is the use of 2-mode networks [2], in which relationships are shaped between two different types of nodes, e.g., users and items. A relatively unexplored area of research concerns the usage of multimode (or n-mode) networks, in which vertices can be of three or even more types. To offer an example from the tourism domain, consider a hotel recommender that models the relationships that appear between users, hotels, and hotel amenities. Furthermore, the internal representation of such networks in usually not visible to users. This can happen for various reasons, ranging from privacy concerns to the inherent difficulty of creating a meaningful illustration of the structure of complex networks.

The purpose of this short paper is to advance the state of the art in two ways. First, we propose a concept of a multimode network for representing users, hotels, and hotel properties. We argue that a hotel recommender can exploit multimode networks to generate more suitable suggestions. Second, we examine potential interactive mechanisms that could be developed to facilitate the presentation of the network to users. We believe that having access to additional means of visualizing hotel information would allow users to interact in novel ways with the RS. Furthermore, such novel interaction techniques could play a role in increasing the trustworthiness of recommendations.

In the following sections, we review related work on the usage of multimode networks in RS. We also investigate some of the typical visualization techniques that have been developed so far. Next, we present our conceptual model for a user-item-attribute network using references and examples from the hotel booking domain. Finally, we discuss how allowing users to interact with such a network might introduce novel ways of interacting with RS. Additionally, we open the discussion on whether exploiting this network could play a role in increasing the trustworthiness of recommendations.

RELATED WORK
The emergence of web communities and the sustained growth of the user-generated content available online provides important opportunities for RS. Incorporating social network information has the potential to alleviate cold-start and sparsity problems, which are inherent in typical collaborative filtering approaches [18]. In the field of RS for tourism (and, more generally, for commerce), a major aspect of exploiting user-generated content is the extraction of topics and sentiments from reviews, e.g., to create richer user models [12]. Combining social networks analysis techniques, such as community detection and visualizing techniques, with RS is, therefore, a worthwhile direction for continued research [10]. A survey of current state-of-the-art

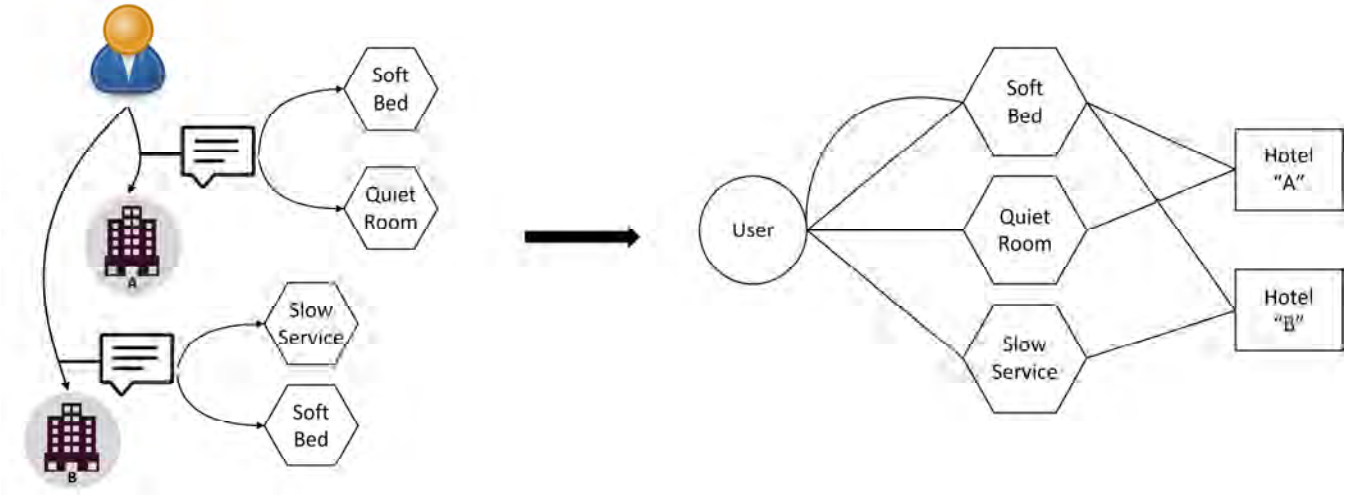


Figure 1 – Example of how a simple multimode network (right) containing three types of vertices—users (circle), topics (hexagon), and hotels (rectangle)—could be generated from crawled user-generate content (left). In our conceptual model, users and hotels are connected through topics, instead of directly.

methods in both fields as well as existing challenges are highlighted, for example, in [18].

Exploiting 2-mode networks for improving the quality of recommendations has already been achieved in various domains. One such approach was used to derive similarity information between artists and songs by exploiting data collected from a music file-sharing network [14]. The information was then used to build a 2-mode graph of users and songs, which was subsequently used to recommend new artists. Citation networks are also a frequent focus of research [9]. The usage of multimode networks for recommendations is less frequent [17]. Linking users to items via tags to improve recommendations is of definite interest in the RS community [15]. Still, tripartite graphs containing users, items, and tags have been studied more in the context of information retrieval [3]. Alternate approaches, for example using multiple 1-mode networks (as opposed to multimode networks) for generating article recommendations, have also been proposed [20].

The problem of visualizing complex networks employed in RS is also receiving increasing attention. Previous work has been done, for instance, on facilitating the exploration of recommended items by combining different entities (users, tags, and agents) [16]. Although there is ongoing research on visualization techniques for tripartite networks [7], it appears that such methods are usually not applied in a systematic way in the context of RS. We argue that multimode networks should not be used solely for enhancing the generation of recommendations. Providing means to visualize the network as well as mechanisms for interacting with it could increase the transparency and control of the RS [16]. The lack of transparency exhibited by many modern RS is frequently cited as having a detrimental effect on the perceived trustworthiness of such systems [6]. Trust is especially relevant in the context of hotel recommending, where the risk associated with poor choices is often higher. Visualizing the underlying factors used to generate recommendations could, for example, allow developers to embed *trust cues* (i.e. interface elements that allow the user to determine the reliability of the presented information) into the presentation layer [13].

CO-STAYING NETWORK

Based on our review of the literature, there appears to be a research gap with respect to the usage of multimode networks for RS. At the same time, the goal of providing interactive mechanisms for allowing users to explore the underlying graph has been studied less frequently.

Example Domain and Dataset

Hotel booking is a domain in which the trustworthiness of recommendations is especially important. More generally, while selecting the domain we considered three aspects: 1) The choice should carry a substantial amount of risk for the user; 2) the items should have a reasonable set of attributes that need to be considered; and 3) there should be a large body of user-generated content available, in the form of reviews, photos, tags, and ratings, that can be leveraged in the presentation. Because of the first criterion, we decided against using the more common domain of movie recommendations. Hotel booking, on the other hand, fulfils all three conditions.

We crawled metadata and overall 838,780 user reviews for 11,544 hotels located in five major European cities from Booking.com[1]. This real-world dataset ensures access to a representative and interesting subset of hotels, covering as many types of amenities as possible. Furthermore, it features a diverse set of reviews contributed by various types of travelers—and who are travelling for different purposes—

[1] http://www.booking.com/

thereby maximizing the variety of user opinions and arguments. A characteristic of Booking.com is that all reviews on its site are *verified*, meaning they are written by people who have stayed in those hotels. This feature reduces the number of fake reviews.

Conceptual Model

Our conceptual model, to which we will refer in the following sections as a "co-staying network", has three types of vertices: users, topics (i.e. hotel attributes), and hotels. In contrast to a typical 2-mode network, users are not linked directly to the hotels in which they stayed. Instead, an edge is first created between the user and a topic. A second edge then links the topic to the hotel for which the review was submitted (Figure 1). This process is repeated iteratively for each topic in a review as well as for all user-contributed reviews. If a review contains references to more than one topic (as is often the case in hotel reviews), there will be several paths between its author and the hotel, each passing through a topic. Furthermore, if a user mentioned the same topic in more than one review, a separate pair of edges will be created for each instance.

The crawled user reviews in the dataset are mined to extract topics. First, we identify attribute-value pairs such as "comfortable bed" in the reviews. Then, these pairs are merged with others that have the same meaning, e.g., "comfy bed". Next, using sentiment analysis, pairs are classified as positive or negative hotel properties. Finally, pairs that describe properties related to, for example, a hotel room, are classified and clustered together. Broadly, topics are divided into three main categories: room attributes (e.g., bed, shower, minibar), hotel attributes (e.g., location, swimming pool, parking), and hotel services (e.g., breakfast, Wi-Fi quality, friendliness of staff). The complete procedure is described in detail in [4]. Topic disambiguation techniques are employed to prevent duplication and correct spelling errors. For example, the terms "wi-fi", "wifi", "wireless network", "wireless internet", or "wifus" (an incorrect spelling that is encountered relatively often in user reviews) should all be filed under a single attribute, e.g., "Wi-Fi".

The user's sentiment regarding each topic is classified with respect to its *polarity* (i.e. positive, neutral, or negative) and *strength*. These two attributes are normalized and encoded in the edges as a single value in the interval [-1,1]. Values closer to the left side of the interval are indicative of strong negative sentiments about a topic (e.g., "terrible breakfast"). Similarly, higher positive values are associated with topics that a user has praised in a review (e.g., "wonderful location").

If a hotel is part of a chain, our model also allows the creation of additional "soft links" from a topic to the rest of the hotels in that chain. This is based on the premise that these hotels share many characteristics—though not all. Thus, it is likely (and typically advertised as such by hotel brands) that travelers would have comparable experiences and access to similar amenities in every location that is part of a franchise.

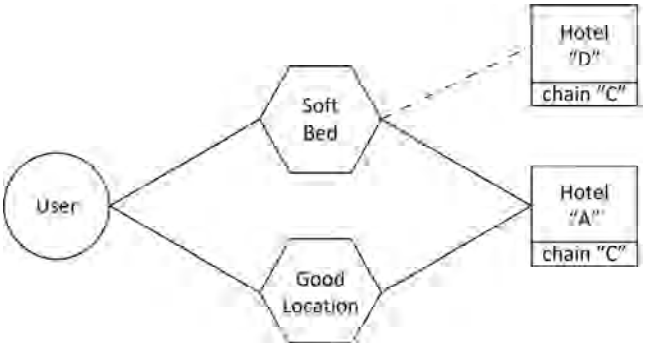


Figure 2 – Example of a soft link from a topic to a hotel that is part of the same chain as the hotel for which the review was written. Soft links can only be created for specific topics.

This approach lets the RS exploit user opinions about selected topics when suggesting recommendations that might otherwise not have enough reviews. Among the topics that could be shared in this way are those concerning room layout and furnishings, breakfast, and hotel facilities. Topics such as those related to the hotel's location or the service quality, on the other hand, would not be shared between hotels (Figure 2).

Clusters can be identified for each type of vertex in the co-staying network. Topics can be characterized by their overarching category (i.e. hotel, room, or service). Furthermore, by analyzing travel and review patterns, additional relationships between hotel topics could be identified. Hotels can be clustered based on whether they are part of a chain as well as by looking at the most common topics that characterize them. Finally, user similarity measures can be extended to include, in addition to demographic data (e.g., age range, country of origin) and rating behavior, also *experience* (e.g., based on the number of contributed reviews, frequency of travel, and the types of hotels visited). Preliminary work on these aspects has already been reported in [1].

The proposed model aims to improve upon traditional approaches to hotel recommending by creating stronger links between users, hotels, and hotel attributes. This would allow users to explore, in addition to the details of the recommended items, the public profiles and preferences of those guests whose reviews were exploited for generating the recommendations. Various forms of interactive mechanisms could be developed to facilitate this kind of exploration.

Interactive Mechanisms

The initial focal point of the network would be the hotel that is being recommended. Users might ask themselves, "What do people talk about when reviewing this hotel?" The most common topics (ideally tailored to fit the user's interests) could be shown radiating from the hotel. These should be clustered by category. Edges would connect these topics to the most representative users. Ensuring that the co-staying network remains accessible to users of the RS is a non-trivial task. One mechanism could involve providing sufficient

criteria to filter topics and users to avoid information overload. Furthermore, the proportion of the co-staying network that is visible to the user should also be controlled, for example by implementing a "zoom in / zoom out" design pattern.

Users should also be able to refocus the network based on their interests or goals. The system should allow seamless transition between vertices, for example between hotels, from a hotel to a topic, or from a topic to the users who referenced it in their reviews. Interacting with a topic should bring up the review snippets in which it is mentioned. The user could then expand reviews that look promising to read them fully; alternatively, less interesting snippets could be hidden completely.

Trust in online reviews has been shown to depend on the credibility of the source [19]. Thus, users should be afforded the possibility to explore the public profiles of reviewers that have contributed opinions about topics of interest. Public profiles might contain information about contributed reviews, visited hotels, frequently-mentioned topics, as well as any demographic data that users may want to share (e.g., purpose of travel, typical number of nights spent in hotels, average price paid). Trustworthiness cues embedded in the personal profiles could help users decide whether to consider—or not—the comments left by other users in the network regarding a certain recommendation. We expect that this might lead to a better calibration between the user's trust in the RS and the system's actual trustworthiness [5].

DISCUSSION AND FUTURE WORK

Including hotel topics as an additional type of vertex in the network is expected to allow the RS to identify, with greater accuracy, which items users are likely to find attractive based on the attributes mentioned in their reviews as well as in reviews left by similar users [3,11]. At the same time, the system could extract and present, for each recommended item, the experiences of other people who are interested in the same combination of topics as the current user. The co-staying network might further facilitate the discovery of users with similar preferences and who have already stayed in the same hotel or in hotels with similar characteristics to the one that a person is considering. We believe that this could thus introduce novel ways of interacting with RS. For example, someone who has a strong preference for soft beds would be able to explore the opinions of other travelers who share this preference. Taking this a step further, the user might then ask herself, "What *other* preferences do such people have?" Personalizing the presentation to facilitate the exploration of similar people's expectations could allow users to discover new preferences that they would not have considered otherwise. A prototype implementation of the presented co-staying concept and interactive mechanisms is currently under development.

We believe that our co-staying network concept has the potential to increase the transparency of the RS by providing visual clues as to why certain recommendations are presented. Moreover, the proposed interactive mechanisms are meant to increase user control over the presentation of the recommended items. Thus, the trustworthiness of the recommendation should increase [6]. Developing evaluation criteria for measuring the effects of these novel interactions on the trustworthiness of the recommendations is planned for future work.

Our approach should also alleviate, to some extent, the data sparsity problem by sharing topics—and, therefore, user opinions—about hotels that are part of a chain. However, this will continue to remain an issue in the case of isolated hotel vertices, for which not enough user-generated information is available from the network. This aspect will be investigated in future work.

ACKNOWLEDGMENTS

This work is supported by the German Research Foundation (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

REFERENCES

1. Barbu, C.-M. & Ziegler, J. User Model Dimensions for Personalizing the Presentation of Recommendations. In *Proc. IntRS '17*. ACM (to appear).
2. Borgatti, S. P. & Everett, M. G. Network analysis of 2-mode data. *Social networks 19*, 243–269 (1997).
3. Clements, M., de Vries, A. P., & Reinders, M. J. Optimizing single term queries using a personalized Markov random walk over the social graph. In *ECIR Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR '08)*, Springer (2008).
4. Feuerbach, J., Loepp, B., Barbu, C.-M., & Ziegler, J. Enhancing an Interactive Recommendation System with Review-based Information Filtering. In *Proc. IntRS '17*. ACM (to appear).
5. Harman, J. L., O'Donovan, J., Abdelzaher, T., & Gonzalez, C. Dynamics of human trust in recommender systems. In *Proc. RecSys '14*, 305–308, ACM (2014).
6. Konstan, J. A. & Riedl, J. Recommender systems: From algorithms to user experience. *User Mod. and User-Adapted Interaction 22*, 1-2, 101–123 (2012).
7. Lambiotte, R. & and Ausloos, M. Collaborative tagging as a tripartite network. In *Proc. ICCS '06*, 1114–1117, Springer-Verlag Berlin Heidelberg (2006).
8. Linden, G., Smith, B., & York, J. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1), 76–80. IEEE (2003).
9. Luong, N. T., Nguyen, T. T., Hwang, D., Lee, C. H., & Jung, J. J. Similarity-based Complex Publication Network Analytics for Recommending Potential Collaborations. *J.UCS*, 21(6), 871–889 (2015).
10. Maivizhi, R., Sendhilkumar, S., & Mahalakshmi, G. S. A Survey of Tools for Community Detection and

Mining in Social Networks. In *Proc. ICIA '16*, 71–79, ACM (2016).

11. McAuley, J. & Leskovec, J. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proc. RecSys '13*, 165–172, ACM (2013).
12. Rossetti, M., Stella, F., & Zanker, M. Analyzing user reviews in tourism with topic models. *Information Technology & Tourism*, 16(1), 5–21 (2016).
13. Sacha, D., Senaratne, H., Kwon, B. C., Ellis, G., & Keim, D. A. The role of uncertainty, awareness, and trust in visual analytics. *IEEE Trans. Vis. Comp. Graph.*, 22-1, 240–249 (2016).
14. Shavitt, Y. & Weinsberg, U. Song clustering using peer-to-peer co-occurrences. In *Proc. ISM '09*. 11th IEEE International Symposium on Multimedia, 471–476, IEEE (2009).
15. Sen, S., Vig, J., & Riedl, J. Tagommenders: connecting users to items through tags. In *Proc. WWW '*09, 671–680, ACM (2009).
16. Verbert, K., Parra, D., Brusilovsky, P., & Duval, E. Visualizing recommendations to support exploration, transparency and controllability. In *Proc. IUI '13*, 351–362, ACM (2013).
17. Wasserman, S. & Faust, K. Social Network Analysis: Methods and Applications. Cambridge Univ. Press (1994).
18. Xu, G., Wu, Z., Zhang, Y., & Cao, J. Social networking meets recommender systems: survey. *International Journal of Social Network Mining*, 2(1), 64–100 (2015).
19. Yoo, K. H., Lee, Y., Gretzel, U., & Fesenmaier, D. R. Trust in travel-related consumer generated media. *Information and communication technologies in tourism 2009*, 49–59, Springer (2009).
20. Zhou, D., Zhu, S., Yu, K., Song, X., Tseng, B. L., Zha, H., & Giles, C. L. Learning multiple graphs for document recommendations. In *Proc. WWW '08*, 141–150. ACM (2008).

Group Recommender Systems in Tourism: From Predictions to Decisions

Tom Gross
University of Bamberg
Kapuzinerstr. 16, 96047 Bamberg
Germany
tom.gross@uni-bamberg.de

ABSTRACT

Recommender systems help users to identify goods or services—typically by offering suitable items from a broad range of alternatives. They have successfully spread into many domains. Tourism is a domain that has a huge potential for simplifying selections and decisions (e.g., on destinations; on itineraries; on accommodation; on cultural activities). In this position paper I discuss how groups of tourists can benefit from group recommender systems and give some examples.

CCS CONCEPTS

• **Human-centered computing** → Collaborative and social computing devices

KEYWORDS

Recommender Systems, Group Recommender Systems, Decision Process, Tourism Recommender Systems

1 INTRODUCTION

The increasing diversity of information, goods, and services offers consumers a huge choice. At the same time finding the preferred item can be challenging. Recommender systems help users making choices by offering suitable items. They have spread into many domains (e.g., book recommendations; music and movie recommendations) [9].

Tourism is a domain with a huge potential for recommender systems to help users to reduce the complexity of planning and deciding, since 'planning a vacation usually involves searching for a set of products that are interconnected (e.g., transportation, lodging, attractions) with limited availability, and where contextual aspects may have a major impact (e.g., spatiotemporal context)' [7].

Group recommender systems support groups of users who want to share information, experiences, or products. Private travelling and touristic activities often happen in pairs or groups: people travel with a partner, people travel with family, people travel to meet friends, but also in business travelling colleagues might travel together or travel to meet working partners or colleagues. Here, group recommender systems are particularly suited, since 'a group recommender is more appropriate and useful for domains in which several people participate in a single activity' [8, p. 199].

Decision making is a core aspect of recommender systems, since the basic assumption is that the system provides suggestions helping users to make informed decisions [9]. For instance, Jameson et al. have identified diverse patterns of humans making a choice [5]. Decision making has also been discussed in the context of group recommender systems, but it has been pointed out that 'only a few studies that concentrate on decision/negotiation support in group recommender systems' [2, p. 30].

In this position paper I share two examples for our own work on group recommender systems: the AGReMo process model for recommendation and decision processes; and the MTEatSplore interactive tabletop applications for groups.

2 THE AGREMO PROCESS MODEL

The AGReMo (Ad-hoc Group Recommendations Mobile) process model was conceived to serve as a blueprint for our group recommender systems that aim to support the full cycle of a recommender process starting with a preparation, followed by a decision, and leading to action. Since it has already been published elsewhere [1], we just quickly glance at it.

As Figure 1 shows, AGReMo consists of three principal phases:

The *Preparation Phase* kicks off the process by collecting all the required data that are needed to later estimate the predictions and generate recommendations. Each group member creates a personal profile. In our case the process model originated from a group recommender systems for movies, so the individual users created a profile and rated movies that they had already seen. After that the group members meet and elect an agent who interacts with the group recommender system (i.e., the assumption is that the whole group meets face-to-face and therefore only needs one system). Then the group can optionally specify group preferences and set preferred parameters. In our case of movie recommendations the group can pre-select cinemas and movies in the region. The group members can furthermore also optionally specify vote weights (i.e., the default was that all group members have equal influence on the recommendation generation, but the group can assign stronger weights to a member, for instance, as a courtesy or due to different levels of expertise). The active agent then requests recommendations, and the system generates group recommendations.

In the *Decision Phase* the group members receive the recommendations with the best prediction on top. The group

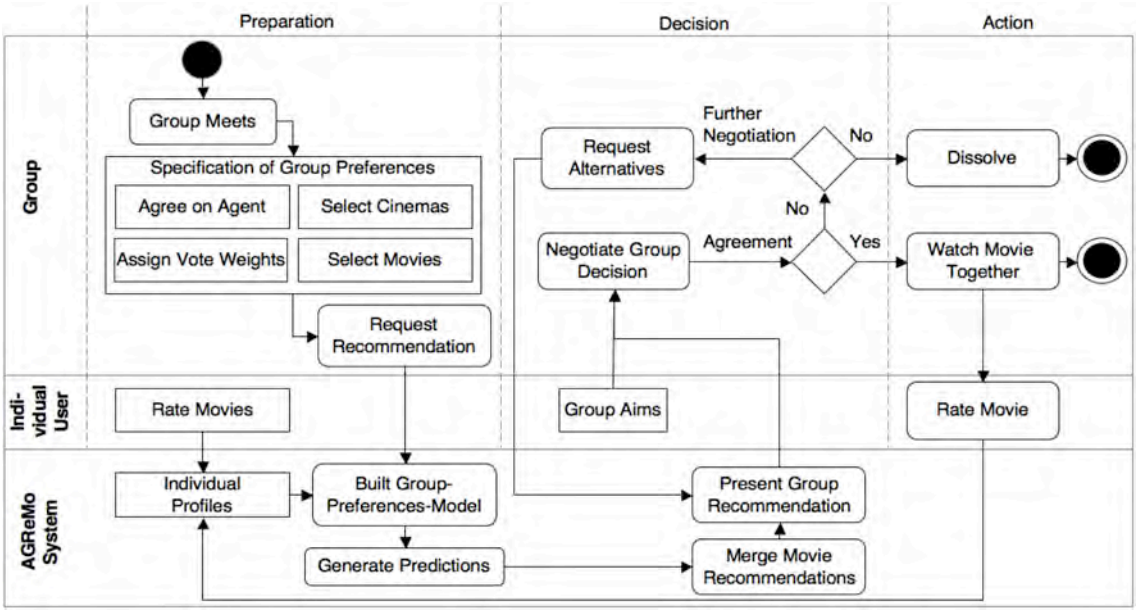


Figure 1. The AGReMo process model. Source: [1].

recommendations are ranked according to the least misery aggregation strategy (i.e., maximising the minimal prediction in the group) [6]. The group can optionally retrieve details for each recommendation, and can also go through suggestions with lower recommendations. They can discuss face-to-face and eventually come to a conclusions.

In the *Action Phase* the group would go to the cinema together and each member is then asked to rate the watched movie to further develop their own profile. The group might also dissolve if no consensus can be reached.

This model was then instantiated in multiple apps (e.g., on app for Android; another app for iOS) and explored with users and based on real-time movie data that were retrieved from our project partner moviepilot [4].

3 THE MTEATSPLORE INTERACTIVE TABLETOP APPLICATION FOR GROUPS

In a different project on a group recommender system we explored the suitability and affordances of interactive tabletops for supporting groups of users in choosing from a set of recommendations generated and presented by the tabletop app.

Here the primary focus was on a concept for the user interaction and user interface for the group decision phase. We started by developing paper prototypes that allows each team member to pick their personal favourite restaurant and to suggest it to the group through a drag-and-drop gesture towards the centre of the table. The table then clusters and aggregates and counts nominations as well as allows the group members to drill down for textual as well as visual background information for the respective restaurant. Figure 2 shows a scenario of the multi-user multi-touch interaction with the restaurant recommendations. Further details on the design process and results can be found it [3].

Figure 2. MTEatSplore scenario showing the multi-user multi-touch interaction with the restaurant recommendations. Source: [3].

4 CONCLUSIONS

In this position paper I have suggested that in the tourism domain it is often *groups* of users who travel together or meet during trips and can benefit from group recommender systems that suggest items of information, services, or goods that are relevant to the whole group. Group recommender systems in tourism face similar

challenges and can benefit from contributions and solutions from other domains.

In the Workshop on Recommenders in Tourism at the 11th ACM Conference on Recommender Systems I would love to discuss ideas and concepts for future work on the whole process of group recommender systems—including technical aspects on how to generate recommendations that reach broad acceptability as well as conceptual aspects of group decision making based on group interaction with recommendations.

ACKNOWLEDGMENTS

I would like to thank the members of the Cooperative Media Lab. Part of the work has been funded by the German Research Foundation (DFG GR 2055/2-1). Thanks to the anonymous reviewers for valuable comments.

REFERENCES

[1] Beckmann, C. and Gross, T. AGReMo: Providing Ad-Hoc Groups with On-Demand Recommendations on Mobile Devices. In Proceedings of the European Conference on Cognitive Ergonomics - ECCE 2011 (Aug. 24-26, Rostock, Germany). ACM, N.Y., 2011. pp. 179-183.

[2] Delic, A., Neidhardt, J. and Nguyen, T.N. Resaerch Methods for Group Recommender Systems. In Workshop on Recommenders in Tourism - RecTour 2016; Co-Located with the 10th ACM Conference on Recommender Systems - RecSys 2016 (Sept. 15, Boston, MA). 2017. pp. 30-37.

[3] Fetter, M. and Gross, T. Engage! Empower! Encourage! - Supporting Mundane Group Decisions on Tabletops. In Proceedings of the 2nd International Conference on Distributed, Ambient, and Pervasive Interactinos - DAPI 2014 (June 22-27, Crete, Greece). Springer-Verlag, Heidelberg, 2014. pp. 329-336.

[4] Gross, T. Supporting Informed Negotiation Processes in Group Recommender Systems. i-com - Journal of Interactive Media 14, 1 (Jan. 2015). pp. 53-61.

[5] Jameson, A., Willemsen, M.C., Felfernig, A., de Gemmis, M., Lops, P., Semeraro, G. and Chen, L. Human Decision Making and Recommender Systems. In Ricci, F., Rokach, L. and Shapira, B., eds. Recommender Systems Handbook. (2nd ed.). Springer-Verlag, Heidelberg, 2015. pp. 611-648.

[6] Masthoff, J. Group Recommender Systems: Combining Individual Models. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P.B., eds. Recommender Systems Handbook. Springer-Verlag, Heidelberg, 2011. pp. 677-702.

[7] Neidhardt, J., Fesenmaier, D., Kuflik, T. and Woerndl, W. Workshop on Recommenders in Tourism. https://recsys.acm.org/recsys17/rectour/, 2017. (Accessed 22/6/2017).

[8] O'Connor, M., Cosley, D., Konstan, J.A. and Riedl, J. PolyLens: A Recommender System for Groups of Users. In Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work - ECSCW 2001 (Sept. 16-20, Bonn, Germany). Kluwer Academic Publishers, Dordrecht, 2001. pp. 199-218.

[9] Ricci, F., Rokach, L. and Shapira, B. Introduction to Recommender Systems Handbook. In Ricci, F., Rokach, L. and Shapira, B., eds. Recommender Systems Handbook. (2nd ed.). Springer-Verlag, Heidelberg, 2015. pp. 1-34.

