RecTour 2019

Workshop on Recommenders in Tourism

Copenhagen, Denmark, September 19th, 2019

Proceedings

Edited by

Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik,

Markus Zanker and Catalin-Mihai Barbu

Co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019)

The ACM Conference Series on Recommender Systems

Copyright and Bibliographical Information

This volume is published by Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, Markus Zanker and Catalin-Mihai Barbu.

Published online at http://ceur-ws.org/Vol-2435/.

Proceedings of the Workshop on Recommenders in Tourism (RecTour 2019), held in conjunction with the 13th ACM Conference on Recommender Systems (RecSys 2019), September 16th - 20th, 2019, Copenhagen, Denmark, https://recsys.acm.org/recsys19/.

Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, Markus Zanker and Catalin-Mihai Barbu (editors).

Further information about the workshop can be found at: http://www.ec.tuwien.ac.at/rectour2019/

Preface

This volume contains the contributions of the Workshop on Recommenders in Tourism (RecTour), organized in conjunction with the 13th ACM Conference on Recommender System (RecSys 2019), in Copenhagen, Denmark. The proceedings were also published online by CEUR Workshop Proceedings at http://ceur-ws.org/Vol-2435/.

RecTour 2019 focuses on a variety of challenges specific to recommender systems in the tourism domain. This domain offers considerably more complicated scenarios than matching travelers with the presumably best items. Planning a vacation usually involves searching for interconnected and dependent product bundles, such as means of transportation, accommodations, attractions, and activities, with limited availabilities and contextual aspects (e.g., spatio-temporal context, social context, activity sequence, and environment) having a major impact. In addition, travel related products can be considered as emotionally loaded and are thus largely experiential in nature; therefore, decision taking is often not solely based on rational or objective criteria. Therefore, information provisioning at the right time about destinations, accommodations and various further services and possible activities is challenging. Additionally, and in contrast to many other recommendation domains, information providers are usually small and medium sized enterprises (SMEs) that many times do not possess the capacity to implement basic recommender systems. Moreover, there is no single, standard format to house information which might be included in these systems. Last, much of the tourism experience is co-produced, i.e., it occurs during the consumption of the product and interaction with the provider. Therefore, the context of the recommendation is extremely important. Thus given this diversity, building effective recommender systems within the tourism domain is extremely challenging. The rapid development of information and communication technologies (ICT) in general and the web in particular has transformed the tourism domain whereby most travelers rely little on travel agents or agencies. Indeed, recent studies indicate that travelers now actively search for information using ICT in order to compose their vacation packages according to their specific emotionally driven preferences. Additionally when on-site, 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. As such, recommender systems have been suggested as a practical tool for overcoming this information overload. However, those designing tourism-focused recommender systems face huge challenges as the tourism domain is extremely complex.

This workshop brings together researchers and practitioners from different fields (e.g., tourism, recommender systems, user modeling, user interaction, mobile, ubiquitous and ambient technologies, artificial intelligence and web information systems) working in the tourism recommendation domain. The workshop aims to provide a forum for these people to discuss novel ideas for addressing the specific challenges for recommender systems in tourism with the goal to advance the current state-of-the-art in this field. Another goal of the workshop is to identify practical applications of these technologies within tourism settings from the point of view of individual users and user groups, service providers, as well as from additional stakeholders (e.g., destination management organizations). Finally, RecTour 2019 aims to continue the community building processes and discussions started at previous RecTour Workshops, i.e., at RecTour 2016 in Boston, MA, USA, at RecTour 2017 in Como, Italy, and at RecTour 2018 in Vancouver, BC, Canada.

August 2019 Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, Markus Zanker and Catalin-Mihai Barbu

Workshop Committees

Organizers

- **Julia Neidhardt**, TU Wien, Austria
- **Wolfgang Wörndl**, TU München, Germany
- **Tsvi Kuflik**, The University of Haifa, Israel
- **Markus Zanker**, Free University of Bozen/Bolzano, Italy
- **Catalin-Mihai Barbu**, University of Duisburg-Essen, Germany

Program Committee

- **Derek Bridge**, University College Cork, Ireland
- **Amra Delic**, TU Wien, Austria
- **Damianos Gavalas**, University of the Aegean, Greece
- **Ulrike Gretzel**, University of Southern California, USA
- **Daniel Herzog**, TU München, Germany
- **Dietmar Jannach**, TU Dortmund, Germany
- **Themis Mavridis**, Booking.com, Netherlands
- **Philipp Monreal,** trivago, Germany
- **Antonio Moreno**, Universitat Rovira i Virgili, Spain
- **Francesco Ricci**, University of Bozen/Bolzano, Italy
- **Hannes Werthner**, TU Wien, Austria

Workshop Program

14:00 - 15:30 Session 1

- **14:00 – 14:05** Workshop opening
- **14:05 – 14:50** Keynote *Building Useful Recommender Systems for Tourists* by **Francesco Ricci** (Free University of Bozen-Bolzano, Italy)
- **14:50 – 15:05 Sebastian Vallejos, Marcelo Gabriel Armentano and Luis Berdun**: *TourWithMe: Recommending peers to Visit Attractions Together*
- **15:05 – 15:25 David Massimo and Francesco Ricci**: *Users' Evaluation of Next-POI Recommendations*
- **15:25 – 15:30** Poster madness:
  - **Ercan Ezin, Hugo Alcaraz-Herrera and Iván Palomares**: *Balancing Preferences, Popularity and Location in Context-Aware Restaurant Deal Recommendation: A Bristol Case Study*
  - **Ramon Hermoso, Sergio Ilarri and Raquel Trillo-Lado:** *Re-CoSKQ: Towards POIs Recommendation Using Collective Spatial Keyword Queries*

15:30 - 16:00 Coffee Break and Posters

16:00 - 17:30 Session 2

- **16:00 – 16:15** Poster presentations
- **16:15 – 16:35 Eoin Thomas, Antonio Gonzalez Ferrer, Benoit Lardeux, Mourad Boudia, Christian Haas-Frangii and Rodrigo Acuna Agost**: *Cascaded Machine Learning Model for Efficient Hotel Recommendations from Air Travel Bookings*
- **16:35 – 16:50 Pavlos Mitsoulis Ntompos, Meisam Hejazinia, Serena Zhang and Travis Brady**: *A Simple Deep Personalized Recommendation System*
- **16:50 – 17:05 Leonhard Seyfang and Julia Neidhardt**: *A Framework for Recommender Systems Based on a Finite Multidimensional Model Space*
- **17:05 – 17:20 Linus W. Dietz, Saadi Myftija and Wolfgang Wörndl**: *Designing a Conversational Travel Recommender System Based on Data-Driven Destination Characterization*
- **17:20 – 17:30** Closing discussion

Building Useful Recommender Systems for Tourists

Keynote by Francesco Ricci (Free University of Bozen-Bolzano, Italy)

Abstract

Recommender systems are information search and filtering tools that should provide suggestions for items to be of use to a user. State of the art recommender systems exploit data mining and information retrieval techniques to predict to what extent an item fits the user needs and wants, but often they end up in making obvious and uninteresting suggestions especially in complex domains, such as tourism. In the talk, classical recommender systems ideas and techniques will be introduced and criticised. We will discuss some of the key ingredients necessary to build a useful recommender system for tourist. Hence, we will point out some limitations and open challenges for recommender systems research. We will then present a couple of novel techniques that are leveraging data collected from observation of tourists behaviour to generate more useful individual and group recommendations.

About the speaker

Prof. Dr. Francesco Ricci is full professor and dean of the Faculty of Computer Science, Free University of Bozen-Bolzano (Italy). F. Ricci has established in Bolzano a reference point for the research on Recommender Systems. He has co-edited the Recommender Systems Handbook (Springer 2011, 2015), and has been actively working in this community as President of the Steering Committee of the ACM conference on Recommender Systems (2007-2010). He was previously (from 2000 to 2006) senior researcher and the technical director of the eCommerce and Tourism Research Lab (eCTRL) at ITC-irst (Trento, Italy). From 1998 to 2000 he was system architect in the Research and Technology Department (Process and Reuse Technologies) of Sodalia s.p.a. F.Ricci has participated to several international research projects such as: RECOM (funded by Deutsche Telekom), etPackaging (funded by ECCA), European Tourist Destination Portal (funded by European Travel Commission), Harmoten (funded by IST), DieToRecs (Intelligent Recommendation for Tourist Destination Decision Making, funded by IST). Francesco Ricci is author of more than one hundred fifty refereed publications and, according to Google Scholar, has H-index 51 and around 15,000 citations.

Table of Contents

Long Papers

Short Papers

Poster Papers

Users' Evaluation of Next-POI Recommendations

David Massimo
damassimo@unibz.it
Free University of Bolzano
Italy

Francesco Ricci
fricci@unibz.it
Free University of Bolzano
Italy

ABSTRACT

The performance of a Recommender System (RS) is often assessed offline, by measuring the system accuracy in predicting or reconstructing the observed user ratings or choices. As a consequence, RSs optimised for that performance measure may suggest items that the user would evaluate correct but uninteresting, because lacking novelty. In fact, these systems are hardly able to generalise the preferences directly derived from the user's observed behaviour. To overcome this problem a novel RS approach has been proposed. It applies clustering to users' observed sequences of choices in order to identify like-behaving users and to learn a user behavioural model for each cluster. It then leverages the learned behaviour model to generate novel and relevant recommendations, not directly the users' predicted choices. In this paper we assess in a live user study how users evaluate recommendations produced by more traditional approaches and the proposed one along different dimensions. The obtained results illustrate the differences of the compared approaches, the benefits and the limitations of the proposed RS.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Human-centered computing** → **User studies**.

KEYWORDS

recommender systems, inverse reinforcement learning, clustering, user study

1 INTRODUCTION

The tourism industry grounds on fulfilling the needs, e.g., accommodation and transportation, of people when moving to a place, for leisure or business purposes [14]. In this industry companies offer online to tourists a wide spectrum of services and activities, such as, city tours, accommodations and food services [15]. However, often the set of available options is so rich that choosing suitable ones can be overwhelming. In order to address this problem, ICT practitioners and far-sighted industries started to develop and employ ad-hoc RSs techniques. Nowadays, the business of companies such as Expedia[1], Booking[2] and Kayak[3] is rooted on recommendation technologies.

In fact, recommender systems are software tools that aim at easing human decision making [16]. In the tourism domain some special dimensions of the recommendation process play an important role. First of all, the demand of activities that a tourist may ask varies in the type and quantity in different contexts. For instance, a tourist may prefer to relax in a park on a sunny day while to visit a museum when it is raining. In order to address this type of requests, Context-Aware RSs (CARS) have been developed [1]. Moreover, since individuals typically consume more than a service or perform more than one activity in a single visit to a destination, session- and sequence-aware recommender systems have been introduced [10]. In tourism applications these methods are used to implement next-POI (Point of Interest) recommendation: recommendations for significant places that the user may be interested to visit next, i.e., after she has visited already some other places (the same day or previously).

In a previous research we developed a novel context-aware recommendation technology (here called *Q-BASE*) for suggesting a sequence of items after the users has already experienced some of them. It models with a reward function the "satisfaction" that a Point of Interest, with some given features, provides to a user [8, 9]. This technique learns the reward function by using only the observation of the users' sequences of visited POIs. This is an important advantage, since typically in on-line systems users scarcely provide feedback on the used services or the visited places. The reward function is estimated by Inverse Reinforcement Learning (IRL), a behaviour learning approach that is widely used in automation and behavioural economics [3, 5]. Moreover, since it is hard to have at disposal the full knowledge, or a huge part of the user history of travel related choices, which would be needed to learn the reward function of a single individual, in [8, 9] IRL is instead applied to clusters of users, and a single learned reward function is therefore shared by all the users in a cluster. For this reason we say that the system has learned a generalised, one per cluster, tourist behaviour model, which identifies the action (POI visit) that a user in a cluster should try next. We studied the proposed approach and compared it with popular baseline algorithms for next-item recommendation [4, 7]. In an offline analysis we have shown that a session-based nearest neighbour algorithm (*SKNN*) generates more precise recommendations while *Q-BASE*, our technique, suggests POIs that are more novel and higher in reward. Hence, we conjectured that, in a real scenario, the latter recommendations may be more satisfying for the user.

In this paper we want to verify that hypothesis, i.e, that users like more the recommendations produced by *Q-BASE*. Moreover, we conjecture that an important difference between the *Q-BASE* method and those based on *SKNN* relies in the "popularity bias": *SKNN* tends to recommend items that have been chosen often by the observed users, while *Q-BASE* is not influenced directly by the popularity of the items, but rather by the popularity of their features. Hence, we introduce here two novel hybrid algorithms that are based on *Q-BASE*, but they deviate from *Q-BASE* by using a popularity score: more popular items tend to be recommended more. These two hybrid algorithms are called *Q-POP COMBINED*

[1] www.expedia.com
[2] www.booking.com
[3] www.kayak.com

and *Q-POP PUSH*. They both combine (in a slightly different way) the item score derived from the reward of the item with a score derived from the item popularity in the users' behaviour data set: more often chosen items (popular) receive a larger score. The items with the largest combined scores are recommended.

We have here repeated the offline analysis of the original *Q-BASE* algorithm and compared its performance with the performance of the two above-mentioned hybrid algorithms, and of two kNN algorithms: *SKNN* that recommends next-item to a user by considering her current session (e.g., visit trajectory) and seeking for similar sessions in the dataset; and sequential session-based kNN (*s-SKNN*) that leverages a linear decay function to weight more in the prediction formula the neighbor trajectories that contain the user's last selected item. Repeating the offline analysis was necessary to validate the conjecture that a significant performance difference between *Q-BASE*- and the *SKNN*- based models is due to the popularity bias of KNN methods. We measure the algorithms offline performance in terms of reward, precision and novelty as it was done in [8]. Moreover, we investigate the effect of the above mentioned hybridization of *Q-BASE*; whether this approach can generate recommendations similar to those computed by *SKNN*. To this end, we compare the Jaccard similarity, of the recommendations (sets) produced by Q-BASE and the hybrid variants, with the recommendations produced by SKNN.

The results of the *offline evaluation* confirm our conjecture: hybridizing *Q-BASE* with item popularity, although it reduces novelty, it increases (offline) precision, aproaching the precision of *SKNN*. Moreover, we show that *Q-POP COMBINED* can still achieve a high reward, whereas *Q-POP PUSH* looses some reward but obtains the same precision of *SKNN*. It is worth noting that as the precision of the proposed hybrid models increase, more and more their produced recommendations overlap with those generated by *SKNN*.

The second major contribution discussed in this paper is an interactive online system aimed at assessing with real users the novelty of and the user satisfaction for the recommendations generated by: the original *Q-BASE* model, one of the two hybrid models (*Q-POP PUSH*) and the same *SKNN* baseline used in the previously conducted offline studies. In the online system the users can enter the set of POI that they previously visited (in Florence) and can receive suggestions for next POIs to visit.

By analysing the users evaluations of the POIs recommended in the *online test*, we found a confirmation that *Q-BASE* suggests more novel items while *SKNN*, as well as the proposed hybrid model *Q-POP PUSH*, offers suggestions that the users like more. We conjecture that, since many items suggested by *Q-BASE* are novel for the users, they are difficult to be evaluated (and liked). We further analyse this aspect by considering recommended items that have been evaluated as "liked and novel" by the users. The results show that *Q-BASE* is better than *SKNN* and *Q-POP PUSH* in suggesting novel and relevant items, which we believe is the primary goal of a recommender system.

In conclusion, in this paper we extend the state of the art in next-POI recommender system with the following contributions:

- Two novel models, *Q-POP COMBINED* and *Q-POP PUSH*, that hybridize the IRL model presented in [8] with a score derived from item popularity.
- An offline study where we show that the proposed hybrid models can obtain precisions similar to those obtained by *SKNN* and *s-SKNN*.
- In a user study we show that when the precision of an algorithms is estimated by leveraging the real user feedback as ground truth, rather than by using the standard ML fictional splitting of train/test, *Q-BASE* performs better than *SKNN* and *Q-POP PUSH* in recommending *novel* items that are liked by the user but it is not better in recommending generic items that are liked.

The paper structure is as follows. In Section 2 the most related works are presented. Then, Section 3 describes how the original IRL-based recommendations are generated [8] and introduces two IRL-based hybrid models. Then, we show how the proposed algorithms compares offline against: the original IRL-based model and the KNN baselines. Section 5 introduces the system developed for the user evaluation and the evaluation procedure. Then, we present the evaluation results. Finally, in Section 7 the conclusion and future works of this study are discussed.

2 RELATED WORK

Our research is focussed on behaviour learning and recommender systems that leverage such behaviour models. Our application scenario is tourism: the goal is to support tourists in identifying what POI they could visit next, given their current location and the information about their past visited places.

Processing and analysing sequences of actions in order to understand the user behaviour to support human decision-making has been already explored in previous research. In [10] is proposed a framework for online experience personalization that leverages users interactions (e.g., clicks) in the form of a sequence. The approach is based on pattern mining techniques in order to identify candidate items, which are present in other users' sequences, that are suitable for recommendations. Another pattern-discovery approach applied to tourism is presented in [13]. Here, the authors propose a RS that identifies next-POI to visit relying on users' check-in sequences data. At first, a directed graph is built from the check-in data and then it is used to identify neighbours of a target user given her check-in data. When neighbours are identified, the POIs in their check-in data are scored. The recommended POI is the one with the maximal score.

Other, more general, pattern-discovery methods are described in [4, 7]. Here the authors present nearest neighbour RS approaches that leverage user behaviour logs: session-based KNN (SKNN) and sequence-aware SKNN (s-SKNN). SKNN seeks for similar users in the system stored logs and identifies the next-item to be recommended given the current user log (session). The s-SKNN weights more weight the neighbours sessions containing the most recent (observed) items of the target user sequence. These methods have been applied to different next-item recommendation tasks showing good performance.

The common aspect of pattern-discovery approaches is that they extract common patterns from user behaviour logs and then learn a predictive model for the next most likely observed user action. That said, these approaches are opaque in explaining the predicted

user behaviour, i.e., users' preferences and their action-selection policy.

To fulfil the need of learning an explainable user behavioural model imitation learning is a viable solution. It is typically addressed by solving Markov Decision Problems (MDP) via Inverse Reinforcement Learning (IRL)[12]. Given a demonstrated behaviour (e.g., user actions sequences) IRL models solve the target MDP by computing a reward (utility) function that makes the behaviour induced by a policy (the learning objective) close to the demonstrated behaviour. In [21] the authors developed an IRL approach based on the principle of maximum entropy that is applied in the scenario of road navigation. The approach is based on a probabilistic method that identifies a choice distribution over decision sequences (i.e., driving decisions) that matches the reward obtained by the demonstrated behaviour. This technique is useful to model route preferences as well as to infer destinations based on partial trajectories. In [3] the authors propose an IRL-based solution to the problem of learning a user behaviour at scale. The application scenario is migratory pastoralism, where learning involves spatio-temporal preferences and the target reward function represents the net income of the economic activity. Similarly, in [5] it is proposed a method for computing the reward humans get by their movements decisions. The paper presents a tractable econometric model of optimal migration, focusing on expected income as the main economic influence on migration. The model covers optimal sequences of location decisions and allows for many alternative location choices. All these works, focus on designing a choice model without studying their application to RSs.

In this work we present two variants of the IRL-based recommender system presented in [8]. There is proposed a RS that first learns users behaviour via IRL and then harnesses it to generate next-item recommendations. In an offline evaluation we showed that the approach excels in novelty and reward, whereas, more precise recommendations are generated by *SKNN-based* techniques. In this paper we argue that the ability of pattern-discovery methods to score high in precision is related to the fact that they are discriminative and are influenced by the observed popularity of the items in the training data. Therefore, in order to leverage item popularity also in an IRL model, we extend the it by hybridizing its scoring function (Q function) with item popularity.

3 RECOMMENDATION TECHNIQUES

3.1 User Behavior Modelling

In this paper, user (tourist) behaviour modelling is based on Markov Decision Processes (MDP). A MDP is defined by a tuple (S, A, T, r, γ). S is the state space and, in our scenario, a state models the visit to a POI in a specific context. The contextual dimensions are: the weather (visiting a POI during a sunny, rainy or windy time); the day time (morning, afternoon or evening); and the visit temperature conditions (warm or cold). A is the action space; in our case it represents the decisions to move to a POI. Hence, POIs and actions are in biunivocal relation. A user that is in a specific POI and context can reach all the other POIs in a new context. T is a finite set of probabilities. $T(s'|s, a)$ is the probability to make a transition from state s to s' when action a is performed. For example, a user that visits Museo del Bargello in a sunny morning (state s_1) and wants to visit Giardino di Boboli (action a_1) in the afternoon can arrive to the desired POI with either a rainy weather (state s_2) or a clear sky (state s_3). The transition probabilities may be equal, $T(s_2, a_1|s_1) = 0.5$ and $T(s_3, a_1|s1) = 0.5$. The function $r : S \rightarrow \mathbb{R}$ models the reward a user obtains from visiting a state. This function is unknown and must be learnt. We take the restrictive assumption that we do not know the reward the user receives from visiting a POI (the user is not supposed to reveal it). But, we assume that if the user visits a POI and not another (nearby) one then this signals that the first POI gives her a larger reward than the second. Finally, $\gamma \in [0, 1]$ is used to measure how future rewards are discounted with respect to immediate ones.

3.2 User Behavior Learning

Given a MDP, our goal is to find a policy $\pi^* : S \rightarrow A$ that maximises the cumulative reward that the decision maker obtains by acting according to π^* (optimal policy). The value of taking a specific action a in state s under the policy π, is computed as $Q_\pi(s, a) = \mathbf{E}^{s,a,\pi}[\sum_{k=0}^{\infty} \gamma^k r(s_k)]$, i.e., it is the expected discounted cumulative reward obtained from a in state s and then following the policy π. The optimal policy π^* dictates to a user in state s to perform the action that maximizes Q. The problem of computing the optimal policy for a MDP is solved by reinforcement learning algorithms [18].

We denote with ζ_u a user u trajectory, which is a temporally ordered list of states (POI-visits). For instance, $\zeta_{u_1} = (s_{10}, s_5, s_{15})$ represent a user u_1 trajectory starting from state s_{10}, moving to s_5 and ending to s_{15}. With Z we represent the set of all the observed users' trajectories which can be used to estimate the probabilities $T(s'|s, a)$.

Since, typically users of a recommender system scarcely provide feedback on the consumed items (visited POIs), the reward a user gets by consuming an item is not known. Therefore, the MDP, which is essential to compute the user policy, cannot be solved by standard Reinforcement Learning techniques. Instead, by having at disposal only the set of POI-visit observations of a user (i.e., the users' trajectories), a MDP for each user could be solved via Inverse Reinforcement Learning (IRL) [12]. In particular, IRL enables to learn a reward function whose optimal policy (the learning objective) dictates actions close to the demonstrated behavior (the user trajectory). In this work we have used Maximum likelihood IRL [2].

3.3 Clustering Users with Similar Behavior

Having the knowledge of the full user history of travel related choices, which would be needed to learn the reward function of a single individual, is generally hard to obtain. Therefore, IRL is here applied to clusters of users (trajectories) [8, 9]. This allows to learn a reward function that is shared by all the users in a cluster. Hence, we say that the system has learned a generalized tourist behavior model, which identifies the action (POI visit) that a user in a cluster should try next.

Clustering the users' trajectories is done by grouping them according to a common semantic structure that can explain the resulting clusters. This is accomplished by employing Non Negative Matrix Factorization (NMF) [6]. NMF extracts topics, i.e., lists of words, that describe groups of documents. Therefore, in order to

apply NMF, we build a document-like representation of a user trajectory that is based on the features (terms) that describe the states visited in a trajectory. Hence, a document-like representation is build for each trajectory in the set Z.

3.4 Recommending Next-POI visits

Here we propose two new next-POI recommendations techniques, *Q-POP COMBINED* and *Q-POP PUSH*, that extend the pure IRL-based *Q-BASE* model, already introduced in [8] (where it was called CBR).

Q-BASE. The behavior model of the cluster the user belongs to is used to suggest the optimal action this user should take next, after the last visited POI. The optimal action is the action with the highest Q value in the user current state [8].

Q-POP COMBINED. In order to recommend more popular items, we propose to hybridise the generalized tourist behavior model learnt for the cluster to which the user belongs to with the item popularity. In particular, given the current state s of a user, for each possible POI-visit action a that the user can make, we apply the following transformation $Q'(s,a) = \frac{Q(s,a)}{\Sigma_i^{|A|} Q(s,a_i)}$ and then we multiply $Q'(s,a)$ by the probability that a POI appears in a user trajectory (in a given data set Z). The result of the multiplication is a distribution that is used to sample the next-POI visit action recommended to the user.

Sampling from a distribution derived from functions composition is widely done in simulation [17]. The approach tries to simulate the decision making process of a user that has all the elements to decide how to act next, i.e., she knows the reward of her future action (the Q values), but she is also biased to select popular items. We conjecture that this method recommends more popular items that have a large reward as well.

Q-POP PUSH. The second hybrid recommendation method introduces even a higher popularity bias to the recommendations generated by *Q-BASE*. We conjecture that it can obtain even a better precision than *Q-POP COMBINED*, closer to the precision of the *SKNN*-based methods. *Q-POP PUSH* scores the visit action a in state s as following:

$$score(s,a) = (1+\beta^2)\frac{Q(s,a)\cdot pop(a)}{(Q(s,a) + pop(a)\cdot\beta^2)}$$

This is the harmonic mean of $Q(s,a)$ and $pop(a)$, which is the scaled (i.e., min-max scaling) counts $c_Z(a)$ (in the data set Z) of the occurrences of the POI-visit corresponding to action a. The harmonic mean is widely used in information retrieval to compute the F1-score. In our case the parameter β was set to 1. The action recommended to the user is the one with the highest score.

4 OFF-LINE ALGORITHM ANALYSIS

4.1 Baselines

We compare here the performance of the recommendations generated by the above mentioned methods with two nearest neighbor baselines: *SKNN* and *s-SKNN*.

SKNN [4] recommends the next-item (visit action) to a user by considering her current session (trajectory) and seeking for similar sessions (neighbourhood) in the data-set. The neighbourhood, i.e., the closest trajectories to the current trajectory, are obtained by computing the binary cosine similarity between the current trajectory ζ and those in the dataset ζ_i: $c(\zeta,\zeta_i)$. Given a set of nearest neighbours N_ζ the score of a visit action a can be computed as:

$$score_{sknn}(a,\zeta) = \sum_{\zeta_n \in N_\zeta} c(\zeta,\zeta_n) 1_{\zeta_n}(a)$$

With 1_{ζ_n} we denote the indicator function: it is 1 if the POI selected by action a appears in the neighbour trajectory ζ_n (0, otherwise). In our data set we cross validated the optimal number of neighbours, and this number is close to the full cardinality of the data set. The recommended actions are those with the highest scores.

s-SKNN [7] extends *SKNN* by employing a linear decay function w_ζ to weight more in the prediction formula the neighbor trajectories that contain the user's last observed visit action and less the earlier visits. The current user trajectory's neighborhood is obtained as in *SKNN*, while the computation of the score of a visit action is as following:

$$score_{s-sknn}(a,\zeta) = \sum_{\zeta_n \in N_\zeta} w_\zeta(a) c(\zeta,\zeta_n) 1_{\zeta_n}(a)$$

For instance, let us say that a_3 is the third observed visit action in the user trajectory ζ (where $|\zeta| = 5$) and that a_3 appears in the trajectory $\zeta_n \in N_\zeta$, then the weight defined by the decay function is $w_{\zeta_n} = 3/5$. Also for *s-SKNN*, the recommended actions are those with the highest scores.

4.2 Evaluation Metrics

The evaluation metrics used to assess the algorithm performance are reward, as defined in [8], precision, novelty and recommendations similarity. Let us denote with $Rec_{u,s}$ a list of recommendations for the user u in state s, and a_o the observed (next) POI-visit (test item). **Reward** measures the average increase in reward that the recommended actions give compared to the observed one:

$$reward(Rec_{u,s}, a_o) = (\sum_{a \in Rec_{u,s}} Q(s,a) - Q(s,a_o))/|Rec_{u,s}|$$

Novelty estimates how unpopular are the recommended visit actions and ranges in $[0,1]$. A POI is assumed to be unpopular if its visits count is lower than the median of this variable in the training set. Let U be the set of unpopular POIs and $1_U(a)$ its indicator function (it is 1 if $a \in U$ and 0 otherwise), novelty is defined as follows:

$$novelty(Rec_{u,s}) = \frac{\sum_{a \in Rec_{u,s}} 1_U(a)}{|Rec_{u,s}|}$$

Let obs_u be the set of observed POI-visit actions in the user u trajectory (test set). The indicator function $1_{obs_u}(a)$ is 1 if $a \in obs_u$ and 0 otherwise. **Precision** is then computed as follows:

$$precision(Rec_{u,s}) = (\sum_{a \in Rec_{u,s}} 1_{obs_u}(a))/|Rec_{u,s}|$$

Finally, we estimate the **Similarity** of two lists of recommendations by computing their Jaccard index. In this study, we compute

the Jaccard index of the recommendations generated by our proposed methods and those generated by *SKNN*. The goal is to verify whether the proposed hybrid methods, which recommend more popular items, improve some of the performances of the pure IRL method Q-BASE and if they recommend items more similar to those recommended by *SKNN*.

4.3 Off-line Study Results

In this study we used an extended version of the POI-visit data-set presented in [11]. It consists of tourist trajectories reconstructed from the public photo albums of users of the Flickr[4] platform. From the information about the GPS position and time of each single photo in an album the corresponding Wikipedia page is queried (geo query) in order to identify the name of the related POI. The time information is used to order the POI sequence derived from an album. In [9] the dataset has been extended by adding information about the context of the visit (weather summary, temperature and part of the day), as well as POI content information (historic period of the POI, POI type and related public figure). In this paper we used an extended version of the dataset that contains 1668 trajectories and 793 POIs.

Trajectories clustering identified 5 different clusters, as in the previous study. In Table 1 we report the performances of Top-1 and Top-5 recommendations for the considered methods. We immediately observe that *SKNN* scores higher in precision, whereas *Q-BASE* suggests more novel and with higher reward items. These results confirm previous analysis [8, 9]. *SKNN* and *s-SKNN* perform very similarly, hence, in this data-set, the sequence-aware extension of *SKNN* seems not to offer any advantage.

When comparing *Q-POP COMBINED* and *Q-POP PUSH* with the two *SKNN*-based methods we found that *Q-POP COMBINED* has a good trade-off between reward and precision. In particular, reward is 4 times (Top-1) the reward of both *SKNN* and *s-SKNN* while precision increases considerably with respect to *Q-BASE*. The same is observed for Top-5 recommendations. But novelty is penalised by the popularity bias of this method.

By looking at the performance of *Q-POP PUSH* we can confirm our study conjecture: a stronger popularity bias enables the algorithm to generate recommendations that are more precise and in particular the precision of *Q-POP PUSH* is equal to that of *SKNN* and *s-SKNN*. But, as expected, reward and novelty are penalised.

With regard to the similarity (Jaccard index) of the recommendations generated by the proposed methods with those of *SKNN*, we can clearly see that the more the precision increases, the higher the Jaccard index becomes. So, the methods are more precise as they are more similar to *SKNN*.

5 ONLINE USER EVALUATION

We conducted an online user-study in order to measure the users' perceived novelty and satisfaction for the recommendations generated by the *Q-BASE* model, the hybrid model *Q-POP PUSH* and the *SKNN* baseline used in the offline study. We designed an online system which first profiles the user by asking her to enter as many as possible previously visited POIs (in Florence). Then the user is asked to evaluate a list of recommendations generated by the aforementioned three models, without being informed of which algorithm recommends what. The data used by the system to train the models and compute recommendations is the same of the offline study, a catalogue of 793 items.

[4] www.flickr.com

Table 1: Recommendation performance

Models	Q-BASE	Q-POP C	Q-POP P	SKNN	s-SKNN
Rew@1	0.073	0.023	-0.002	-0.007	-0.009
Prec@1	0.043	0.057	0.099	0.109	0.109
Nov@1	0.061	0.029	0.000	0.000	0.000
Jacc@1	0.085	0.106	0.424	-	0.791
Rew@5	0.032	0.017	-0.009	-0.010	-0.010
Prec@5	0.045	0.049	0.060	0.068	0.063
Nov@5	0.122	0.040	0.000	0.000	0.000
Jacc@5	0.061	0.063	0.192	-	0.530

5.1 Online Evaluation System

The interaction with the system unfolds as follow: landing phase; introduction to the experiment and start up questions; preference elicitation phase; recommendation generation and evaluation.

Once the user accesses the website she can select the language (Italian or English) and then, if the user accepts to participate to the experiment, she is asked whether has already been in Florence. If she replies "no" the procedure ends. Otherwise, the user is considered to have some experience of the city and can declare which POIs has already visited. In this case, the preference elicitation phase is supported by a user interface (Figure 1) that enables the user to select as many POIs she remembers to have visited in Florence. The selection can be performed in two non-exclusive modalities. The first one is a lookup bar with auto-completion, while the second is a selection pane that contains the most popular 50 POIs. If the user hovers or taps on an item the system renders a media card presenting content extracted from Wikipedia: a picture and a textual description. When the user selects a POI as visited, this is added to an (editable) list. The selected POIs are meant to build a user profile which is then used to identify the best representative user's trajectory cluster, among the 5 clusters of previously collected training data (the details of this computation are explained in the next section).

Then the system generates a short itinerary (5 POIs) composed by a small sample of the POIs that the users previously declared to have visited (Figure 2). This is the itinerary that the user is supposed to have followed just before asking a recommendation for a new point to visit. We decided to generate a fictitious itinerary because we did not want to ask the user to remember any previous visit itinerary, but we also tried to generate a trajectory that is likely to have been followed (by sampling among the POIs that entered in the profiling stage). By showing a hypothetical itinerary to the user, followed up to the current point, we wanted to reinforce in the user the specific setting of the supported recommendation task: next-POI recommendation.

That said, the recommendation generation and evaluation phase present a user interface that is organized as follows. At the top of the page there is an information box containing the (previously mentioned) hypothetical (5-POIs) trajectory that the user should assume has followed (Figure 2). Below, there is an info box that explains the participant to assume that she has visited the selected

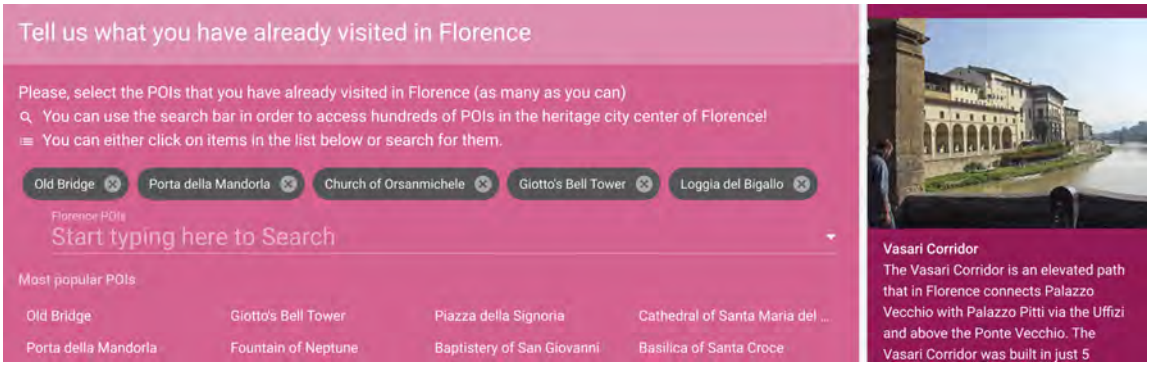

Figure 1: POI selection UI detail.

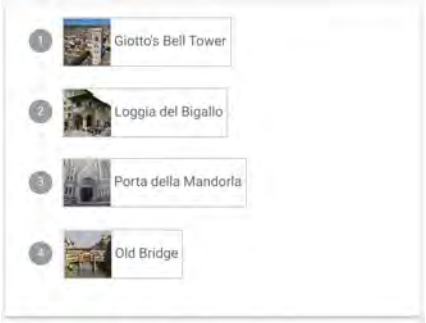

Figure 2: Itinerary detail.

attractions, in the presented order. Finally, the participant is informed that the beneath box (Figure 3) contains a list of POIs that she can visit (recommendations) after the last POI in the itinerary. The user is asked to mark the recommendations with one or more of the following labels: "I already visited it" (eye icon), "I like it" for a next visit (thumb up icon) and "I didn't know it" (exclamation mark icon).

We recruited the experiment participants via social media and mailing lists and we collected over 300 responses of which 202 are from users that visited Florence. After excluding unreliable replies (e.g., survey completed in less than 2 minutes) we counted 158 users. The number of recommended next-POI visits shown to the users is 1119 (approximately three by each of the three methods per user, excluding the items recommended by two or more method simultaneously). Hence on average a user has seen 7.1 recommendations.

5.2 Recommendation List Generation

In order to generate recommendations using *Q-BASE* and *Q-POP PUSH* an online user must be associated to one of the five existing trajectories' clusters. In fact, the user behavioural model is considered to be shared with the other users in the same cluster, and it is learned by using the trajectories already present in the cluster.

Matching a user to a cluster. In order to associate an online user to a pre-existent cluster (among the 5 that we created) we built a tf-idf representation of the POIs (documents) that are in the user

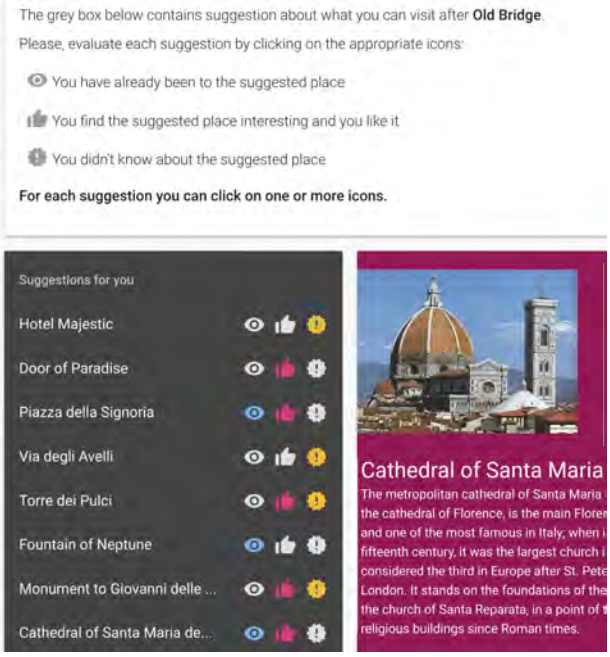

Figure 3: Evaluation. UI detail.

profile and then we run a nearest neighbor classifier where the training data are the existent trajectories in the data set, already classified in the 5 clusters. We assessed the classifier performance by splitting the trajectories data set: 80% of the dataset has been used for training the classifier and the remaining 20% has been used as test set. In a 10-fold cross-validation the classifier showed an accuracy of 0.67. Hence, the quality of this classifier is not very high. This may have penalised both *Q-BASE* and *Q-POP PUSH* in the online study.

5 POIS Fictitious Itinerary. Once the user is associated to a cluster, among all the trajectories in the cluster we identify the trajectory in the cluster with the highest overlap (intersection) with the POIs selected by the study participant (randomly breaking ties). On

the user interface, as we mentioned above, in order to avoid information overload, we show to the user at most 5 items, of her user profile, ordered according to the matched itinerary, found in the matched cluster. The itinerary is shown to the user as her current (hypothesized) sequence of visited POIs in order to evaluate the next-POI recommendations as appropriate or not to complete the initiated itinerary.

Recommendations. Given the fictitious hypothesized itinerary followed by the user so far, next-POI recommendations are independently generated leveraging the algorithms *Q-BASE*, *Q-POP* and *kNN*. Then, from the recommendations generated by the algorithms we filter out (post-filtering) the POIs already in the user profile. This is an important feature of our study: we wanted to suggest POIs that are good for a next visit, i.e., that the user has not yet visited [5]. Moreover, in order to avoid biases in the recommendation evaluation phase we do not reveal to the user which recommendation algorithm has produced which POI recommendation.

Furthermore, to control the "position bias" [19, 20], i.e., the tendency of users to select top positioned items in a list, regardless of their relevance, we aggregate the top-3 suggestions of each algorithm without giving to any algorithm a particular priority. In fact, at first, we (randomly for each user) generate an order that we follow to pick items from the three lists of the top-3 suggestions generated by the three considered algorithms. Then we aggregate the three ranked list by picking up, in turn, the items from the top to the bottom of the sorted lists. For instance, if the generated order is *Q-BASE*, *kNN* and *Q-POP*, then, the aggregated list of recommendations (max length 9) that is shown to the user, contains in the first position the top recommendation of *Q-BASE*, then the top item suggested by *kNN* and then that suggested by *Q-POP*. The same pattern is applied for the remaining positions: the fourth item in the aggregated list is the second best POI suggested by *Q-BASE* and at the fifth and sixth positions are placed the second best POIs suggested by *kNN* and *Q-POP*. In the case a POI is suggested by more than one algorithm, the item is shown only once.

6 RESULTS OF THE ONLINE USER STUDY

The results of the recommendation generation and evaluation phase are shown in Table 2. We show here the probabilities that a user marks as "visited", "novel", "liked" (for a next visit) or both "liked" and "novel" an item recommended by an algorithm. They are computed by dividing the total number of items marked as, visited, liked, novel and both liked and novel, for each algorithm, by the total number of items shown by an algorithm. By construction, each algorithm contributes with 3 recommendations in the aggregated list shown to each user. It is worth stressing that a user marked as "liked" an item that she judged as a good candidate for a next POI visit. Hence, here a "like" is not a generic appreciation of the item, but takes (partially) into account the visit context (what items the user has already visited).

We note that the POIs recommended by *SKNN* and *Q-POP* have the highest probability (24%) that the user has already visited them, and the lowest probability to be considered as novel. *Q-BASE* scores a lower probability that the recommended item be already visited (16%) and the highest probability that the recommended item be novel (52%). This is in line with the offline study where *Q-BASE* excels in recommending novel items.

Considering now the user satisfaction for the recommendations (liked), we conjectured that a high reward of an algorithm measured offline, corresponds to a high perceived satisfaction (likes) measured online. But, by looking at the results in Table 2 we have a different outcome. *Q-BASE*, which has the highest offline reward recommends items that an online user likes with the lowest probability (36%). *Q-POP PUSH* and *SKNN* recommend items that are more likely to be liked by the user (46%).

Another measure of system precision that we computed is the probability that a user likes a novel recommended POI, i.e., a POI that the recommender presented for the first time to the user ("Liked & Novel" in Table 2). We note that this is the primary goal of a recommender system: to enable users to discover items that are interesting for them, not to suggest items that the user likes, but that she is already aware of, or she has already consumed. There is poor utility of such a functionality. In this case, *Q-BASE* (highest reward and lowest precision offline) recommends items that a user will find novel and also like with the highest probability (0.09%), whereas *SKNN* and *Q-POP PUSH* recommends items that the user will find novel and will like with a lower probability(0.08%). We believe that the online computed "Liked & Novel" probability is a better measure of the precision of a RS. In fact, the standard offline estimation of precision, which is computed on the base of an artificial split of the available liked items into train/test is not able to estimate how, not yet experienced items that the recommender suggests may be liked by the user. It is also worth noting the low scores of this metric: it is hard to observe a user that liked a novel item. This aspect is further discussed below.

In order to further study the online user evaluation of the recommended items, we have computed the probability that a user will like recommendations given the fact that she knows the item but has not yet visited it ("Known & Not Visited"), she visited it ("Visited") or the item is "Novel" for her. The results of this analysis are shown in Table 3. The novel POIs recommendations generated by *SKNN* and *Q-POP PUSH* are liked more (20% and 22%) than those produced by *Q-BASE* (17%). We believe that this is because often *Q-BASE* suggests items that are very specific and users may find hard to evaluate them. For instance, *Q-BASE* suggests often "Porta della Mandorla" which is a door of the "Duomo". This POI can be perceived as a niche item and much less attractive than the "Duomo" itself. Moreover, by conducting post-survey activities participants declared that it is difficult to like something that is unknown.

In fact, the probability that a user likes a recommended POI that she has visited tends to be much larger. This probability is 31% and 28% for *Q-POP PUSH* and *SKNN* respectively. Whereas, *Q-BASE* also here performs worse (26%). We think that the performance difference is again due to the fact that both *SKNN* and *Q-POP* tend to recommend popular POIs (easier to judge), whereas *Q-BASE* recommends more "niche" items.

Considering now the probability that a user will like an item that she knows but has not yet visited we see again a similar pattern as before: *Q-POP PUSH* and *SKNN* suggest items that will be liked with a higher probability (81% and 80%) than *Q-BASE* (71%). These

[5] Still some recommendations can be not novel because the user will never declare all the POIS that she visited or she knows in the city.

Table 2: Probability to evaluate a recommendation of an algorithm as visited, novel and liked.

	Q BASE	Q POP	SKNN
Visited	0.165	0.245	0.238
Novel	0.517	0.376	0.371
Liked	0.361	0.464	0.466
Liked & Novel	0.091	0.076	0.082

Table 3: Probability that a user likes a suggested item given that she visited, knew or is unaware of it.

	Q BASE	Q POP	SKNN
P(Liked | Novel)	0.176	0.202	0.222
P(Liked | Visited)	0.256	0.310	0.283
P(Liked | Known & Not Visited)	0.717	0.810	0.806

probabilities are very large. We conjecture that this is because these are popular items that the user has not yet visited. In fact, if we compare the probabilities that a user will like an item given that is novel, visited or known but not yet visited, we see that it is the largest for the latter items (> 70%), it is lower for the visited items (> 26%) and and the lowest for the novel items (< 22%). This reinforce the conjecture that users tends to like items they are familiar with (but they have not yet consumed).

7 CONCLUSION AND FUTURE WORK

In this paper we extend the state of the art in IRL-based next-POI RSs. We started our analysis by hypothesising that users like more the recommendations produced by IRL-models and that the poor offline accuracy of these models, compared to KNN approaches, is due to the total absence of a popularity bias in the recommendation generation. For that reason we designed two new hybrid models that bias the pure IRL-model *Q-BASE* to suggest more popular items: *Q-POP COMBINED* and *Q-POP PUSH*.

We show with an offline experiment that the hybridization of *Q-BASE* results in an increase of precision: *Q-POP PUSH* performs equally to SKNN-based approaches.

With an *online test* we show that the *Q-BASE* model excels in suggesting novel items, whereas *SKNN* and *Q-POP PUSH* suggests items that are "liked" more. We also show that if we consider the combined feedback "liked and novel", i.e., recommendations that are liked and are novel to the user, *Q-BASE* outperforms both *SKNN* and *Q-POP PUSH*. Hence, we show that *Q-BASE* may be able to better accomplish the most important task of a RS for tourism: suggesting relevant POIs that are unknown for a user and also relevant.

We emphasize here that the objective of this research is a next-POI RS that harnesses a generalized tourist behavior model. While in this work we showed the benefits of such a RS through a web-based study we are now conducting a novel user study with real tourists interacting with a system while visiting a destination (South Tyrol)[6].

[6]http://wondervalley.unibz.it
https://beacon.bz.it/wp-6/beaconrecommender/

Another future work direction is the analysis of the users' perception of the recommendations generated by the different algorithms given the possibly different users' knowledge of the target destination.

ACKNOWLEDGMENTS

The research described in this paper was developed in the project Suggesto Market Space in collaboration with Ectrl Solutions and Fondazione Bruno Kessler.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. 2011. Context-Aware Recommender Systems. In *Recommender Systems Handbook*, F. Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 217–253.
[2] M. Babes-Vroman, V. Marivate, K. Subramanian, and M. Littman. 2011. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning - ICML'11*. 897–904.
[3] S. Ermon, Y. Xue, R. Toth, B. Dilkina, R. Bernstein, T. Damoulas, P. Clark, S. DeGloria, A. Mude, C. Barrett, and C. P. Gomes. 2015. Learning Large Scale Dynamic Discrete Choice Models of Spatio-Temporal Preferences with Application to Migratory Pastoralism in East Africa. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Pattern*, 644–650.
[4] D. Jannach and L. Lerche. 2017. Leveraging Multi-Dimensional User Models for Personalized Next-Track Music Recommendation. In *Proceedings of the Symposium on Applied Computing - SAC'17*. 1635–1642.
[5] J. Kennan and J. R. Walker. 2011. The Effect of Expected Income on Individual Migration Decisions. *Econometrica* 79, 1 (2011), 211–251.
[6] D. D. Lee and H. S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.
[7] M. Ludewig and D. Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Model. User-Adapt. Interact.* 28, 4-5 (2018), 331–390.
[8] D. Massimo and F. Ricci. 2018. Harnessing a generalised user behaviour model for next-POI recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. 402–406.
[9] D. Massimo and F. Ricci. 2019. Clustering Users' POIs Visit Trajectories for Next-POI Recommendation. In *Information and Communication Technologies in Tourism 2019, ENTER 2019, Proceedings of the International Conference in Nicosia, Cyprus, January 30-February 1, 2019*. 3–14.
[10] B. Mobasher, H. Dao, T. Luo, and M. Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *Proceedings of the IEEE International Conference on Data Mining - ICDM '02*. 669–672.
[11] C. I. Muntean, F. M. Nardini, F. Silvestri, and R. Baraglia. 2015. On Learning Prediction Models for Tourists Paths. *ACM Transactions on Intelligent Systems and Technology* 7, 1 (2015), 1–34.
[12] A. Ng and S. Russell. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning - ICML '00*. 663–670.
[13] S. Oppokhonov, S. Park, and I. K. E. Ampomah. 2017. Current Location-based Next POI Recommendation. In *Proceedings of the International Conference on Web Intelligence (WI '17)*. ACM, New York, NY, USA, 831–836.
[14] World Tourism Organization. 1995. *Collection of Tourism Expenditure Statistics*. World Tourism Organization (UNWTO).
[15] Revfine.com. 2019. Travel and Tourism Industry; An complete Overview of All Activities. https://www.revfine.com/travel-and-tourism
[16] F. Ricci, L. Rokach, and B. Shapira. 2015. Recommender Systems: Introduction and Challenges. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 1–34.
[17] C. R. P. Robert and G. Casella. 2010. *Introducing Monte Carlo Methods with R*. EU-Nachrichten, Themenheft, Vol. 30. Springer, New York, NY u.a.
[18] R. S Sutton and A. G. Barto. 2014. *Reinforcement Learning: An Introduction (Second edition, in progress)*. The MIT Press.
[19] E. C. Teppan and M. Zanker. 2015. Decision Biases in Recommender Systems. *Journal of Internet Commerce* 14, 2 (2015), 255–275.
[20] X. Wang, M. Bendersky, D. Metzler, and M. Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 115–124.
[21] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - AAAI'08*. 1433–1438.

Cascaded Machine Learning Model for Efficient Hotel Recommendations from Air Travel Bookings

Eoin Thomas*
Antonio Gonzalez Ferrer*
eoin.thomas@amadeus.com
Amadeus SAS
Sophia Antipolis, France

Benoit Lardeux
Amadeus SAS
Sophia Antipolis, France

Mourad Boudia
Amadeus SAS
Sophia Antipolis, France

Christian Haas-Frangii
Amadeus SAS
Sophia Antipolis, France

Rodrigo Acuna Agost
Amadeus SAS
Sophia Antipolis, France

ABSTRACT

Recommending a hotel for vacations or a business trip can be a challenging task due to the large number of alternatives and considerations to take into account. In this study, a recommendation engine is designed to identify relevant hotels based on features of the facilities and the context of the trip via flight information. The system was designed as a cascaded machine learning pipeline, with a model to predict the conversion probability of each hotel and another to predict the conversion of a set of hotels as presented to the traveller. By analysing the feature importance of the model based on sets of hotels, we are able to construct optimal lists of hotels by selecting individual hotels that will maximise the probability of conversion.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**;

KEYWORDS

Recommender systems, machine learning, hotels, conversion.

1 INTRODUCTION

In the United States, the travel industry is estimated to be the third largest industry after the automotive and food sectors and contributes to approximately 5% of the gross domestic product. Travel has experienced rapid growth as users are willing to pay for new experiences, unexpected situations, and moments of meditation [9, 28], while the cost of travel has decreased over time in part due to low cost carriers and the sharing economy. At the same time, traditional travel players such as airlines, hotels, and travel agencies, aim to increase revenue from these activities. The supply side must identify its market segments, create the respective products with the right features and prices, and it has to find a distribution channel. The traveller has to find the right product, its conditions, its price and how and where to buy it. In fact, the vast quantity of information available to the users makes this selection more challenging.

Finding the best alternative can become a complicated and time-consuming process. Consumers used to rely mostly on recommendations from other people by word of mouth, known products from advertisements [20] or inform themselves by reading reviews [6, 18]. However, the Internet has overtaken word of mouth as the primary medium for choosing destinations [23] by guiding the user in a personalized way to interesting or useful products from a large space of possible options.

Many players have emerged in the past decades mediating the communication between the consumers and the suppliers. One type of player is the Global Distribution System (GDS), which allows customer-facing travel agencies (online or physical) to search and book content from most airlines and hotels. Increased conversion is a benefical goal for the supplier and broker as it implies more revenue for a lower cost of operation, and for the traveller, as it implies quicker decision making and thus less time spent on search and shopping activities.

In this study, we aim to increase the conversion rate for hospitality recommendations after users book air travel. In Section 2, the problem is formulated in order to highlight the considerations which separate this work from many recommender system paradigms. Section 3 presents the main techniques and concepts used in this study. In Section 4, a brief overview is given of the industry data used in this study. Section 5 discusses the results obtained for different machine learning models including feature analysis. A discussion of the main outcomes of this study is provided in Section 6.

2 PROBLEM FORMULATION

2.1 Industry background

Booking a major holiday is typically a yearly or bi-yearly activity for travellers, requiring research for destinations, activities and pricing. According to a study from Expedia [12], on average, travellers visit 38 sites up to 45 days prior to booking. The travel sector is characterized by Burke and Ramezani [5] as a domain with the following factors:

- Low heterogeneity: the needs that the items can satisfy are not so diverse.
- High risk: the price of items is comparatively high.
- Low churn: the relevance of items do not change rapidly.
- Explicit interaction style: the user needs to explicitly interact with the system in order to add personal data. Although some implicit preferences can be tracked from web activity and

*Both authors contributed equally to this research.

past history, mainly the information obtained is gathered in an explicit way (e.g. when/where do you want to travel?).
- Unstable preferences: information collected from the past about the user might be no longer trustworthy today.

Researchers have tried to relate touristic behavioural patterns to psychological needs and expectations by 1) defining a characterization of travel personalities and 2) building a computational model based on a proper description of these profiles [27]. Recommender systems are a particular form of information filtering that exploit past behaviours and user similarities. They have become fundamental in e-commerce applications, providing suggestions that adequately reduce large search spaces so that users are directed toward items that best meet their preferences. There are several core techniques that are applied to predict whether an item is in fact useful to the user [4]. With a content-based approach, items are recommended based on attributes of the items chosen by the user in the past [3, 26]. In collaborative filtering techniques, recommendations to each user are based on information provided by similar users, typically without any characterization of the content [19, 24, 25]. More recentely, session-based recommenders have been proposed, where content is selected based on previous activity made by the user on a website or application [17].

2.2 Terminology

In order to clearly define our goal, let us first define some terminology:

- **Hotel Conversion**: a hotel recommendation leads to a conversion when the user books a specific hotel.
- **Hotel Model**: machine learning model trained to predict the conversion probability of individual hotels.
- **Passenger Name Record (PNR)**: digital record that contains information about the passenger data and flight details.
- **Session**: after a traveller completes a flight booking through a reservation system, a session is defined by the context of the flight, the context of the reservation, and a set of five recommended hotels proposed by the recommender system.
- **Session Conversion**: a session leads to a conversion when the user books any of the hotels suggested during the session.
- **Session Model**: machine learning model trained using features related with the session context and hotels, its output is the conversion probability of the session.

The end goal of the recommender system is to increase session conversion. We can estimate the probability of booking of a list of hotels using the session model, and thus we can compare different lists using the session model to determine the one which will maximise the probability of conversion of the session. Note that in this case conversion is defined as a selection or "click" of a hotel on the interface, rather than a booking.

2.3 Hotel recommendations

The content sold through a GDS is diverse, including flight segments, hotel stays, cruises, car rental, and airport-hotel transfers. The core GDS business concerns the delivery of appropriate travel solutions to travel retailers. Therefore, state-of-the-art recommendation engines capable of analysing historical bookings and automatically recommending the appropriate travel solutions need to be designed. Figure 1 shows an outline of the rule-based recommendation system currently in use. After a user books a flight, information related to the trip is sent to the recommender engine.

However, this system does not take into account valuable information such as the context of the request (e.g. where did the booking originate from?), details about the associated flight (e.g. how many days is the user staying in the city?) nor historical recommendations (e.g. are similar users likely to book similar hotels?), which are key assets to fine tune the recommendations.

The problem is novel due to the richness of available data sources (bookings, ratings, passenger information) and the variety of distribution channels: indirect through travel agencies or direct (website, mobile, mailbox). However, it is important to consider that by design, no personally identifiable information (PII) or traveller specific history is used as part of the model, which therefore excludes collaborative-filtering or content-based approaches. The contributions of this work are:

- The combination of data feeds to generate the context of travel, including flights booked by traveller, historical hotels proposed and booked at destination by other travellers, and hotel content information.
- The definition of a 2-stage machine learning recommender tailored for travel context. Two machine learning models are required to build the new recommendation set. The output of the first machine learning algorithm (prediction of the probability of hotel booking) is a key input for the second algorithm, based on the idea of [13].
- The comparison of several machine learning algorithms for modelling the hospitality conversion in the travel industry.
- The design and implementation of a recommendation builder engine which generates the hotel recommendations that maximize the conversion rate of the session. This engine is built based on the analysis of the feature importance of the session model at individual level [29].

3 METHODOLOGY

3.1 Pipeline

Using machine learning and the historical dataset of recommendations, we can train a model which is capable of predicting with high confidence whether a proposed set of recommended hotels leads to a booking.

Once we have fit the model, we can evaluate other combinations of hotels and recommend a list of hotels to the user that maximizes the conversion. Instead of proposing a completely new set of hotels, we decide to modify the existing suggestions given by the existing rule-based system. Our approach, shown in Figure 2, removes one of the initial hotels and introduces an additional one that increases the conversion probability:

We have identified two different ways to select the hotel that is going to be introduced within the set of recommendations:

- We can create and evaluate all possible combinations and choose the one with the highest conversion probability. This means, each time one out of the five hotels from the initial list is removed, and a new one from the pool of hotels is inserted. However, this brute force solution is computationally inefficient and time-consuming (e.g., in Paris this results in

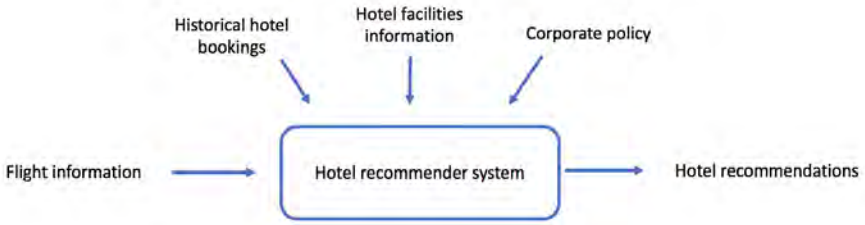

Figure 1: A hotel recommendation system. When a flight booking is completed, the flight details are passed to the hotel recommender engine which selects a set of available hotels for the user based on historical hotel bookings, hotel facilities and a corporate policy check.

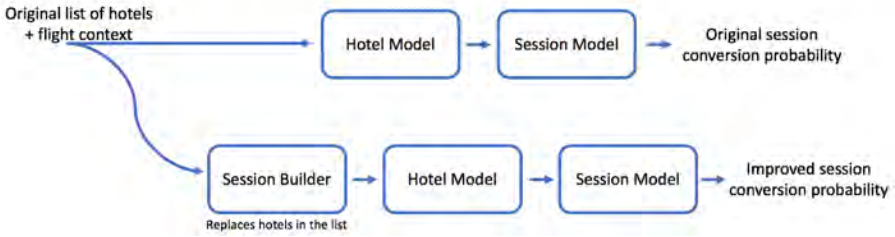

Figure 2: The goal of the system is to improve the probability of conversion. To provide a better set of recommendations, the session builder replaces hotels in the original list.

5*1,653 different combinations for a single swap, the length of the list multiplied by the number of available hotels).

- Alternatively, a hotel from the list of selected hotels can be replaced with an available hotel, based on some criteria. Typically, the criteria might be the price of the hotel room, or the average review score, or a combination of multiple indicators. In this work, the criteria used to optimise the overall list of hotels is determined via feature analysis.

Nevertheless, the last solution presents some challenges that need to be discussed and solved:

(1) How to study the feature importance of complex non-linear models?
(2) How to best interpret the feature importance in an unbalanced dataset?
(3) How many features should be used during the selection process of building an optimal list? Initially, we are facing a multi-objective optimization problem since the choice of a hotel for enhancing the conversion probability might depend on different features. Furthermore, the existence of categorical features makes this optimization even harder. Can we convert it into a univariate optimization problem?

The novelty of this study comes from the use of two related works to address the above points. First, we design a two-stage cascaded machine learning model [13] where the output probabilities of the first model are a new feature of the second one. Second, we interpret the feature importance of the positive instances (i.e. conversions) with a local interpretable model-agnostic (LIME) technique [29]. Thus, we can study the feature importance of particular instances in complex models, allowing the switch from a multi-objective to a univariate optimization problem when one feature is dominant.

3.2 Cascade Generalization

Ensembling techniques consist in combining the decisions of multiple classifiers in order to reduce the test error on unseen data. After studying the bias-variance decomposition of the error in bagging and boosting, Kohavi observed that the reduction of the error is mainly due to reduction in the variance [21]. An issue with boosting is robustness to noise since noisy examples tend to be misclassified and therefore the weight will increase for these examples [2]. A new direction in ensemble methods was proposed by Gama and Brazdil [13] called Cascade Generalization. The basic idea is to use sequentially the set of classifiers (similarly to boosting), where at each step, new attributes are added to the original data. The new attributes are derived from the probability class distribution given by the base classifiers.

There are several advantages of using cascade generalization over other ensemble algorithms:

- The new attributes are continuous since they are probability class distributions.
- Each classifier has access to the original attributes and any new attribute included at lower levels is considered exactly in the same way as any of the original attributes.
- It does not use internal cross validation which affects the computational efficiency of the method.
- The new probabilities can act as a dimensionality reduction technique. The relationship between the independent

features and the target variable are captured by these new attributes.

As will be shown in further sections, this last point is a key aspect of the proposed system, as the probabilities generated by the hotel model can be used to directly select new hotels to include in the recommendation. However, the session model uses aggregated features from the hotel model, and as such an interpretable feature analysis is required to determine how best to select hotels based on their features.

3.3 Interpretability in Machine Learning

Machine learning has grown in popularity in the last decade by producing more reliable, more accurate, and faster results in areas such as speech recognition [16], natural language understanding [8], and image processing [22]. Nevertheless, machine learning models act mostly as black boxes. That is, given an input the system produces an output with little interpretable knowledge on how it achieved that result. This necessity for interpretability comes from an incompleteness in the problem formalisation meaning that, for certain problems, it is not enough to get the solution, but also how it came to that answer [11]. Several studies on the interpretability for machine learning models can be found on the literature [1, 15, 32].

3.4 Local Interpretable Model-Agnostic Explanations (LIME)

In this section, we focus on the work from Ribeiro et al. [29] called Local Interpretable Model-Agnostic Explanations. The Local Interpretable Model-Agnostic Explanations model explains the predictions of any classifier (model-agnostic) in a interpretable and faithful manner by learning an interpretable model locally around the prediction:

- **Interpretable**. In the context of machine learning systems, we define interpretability as the ability to explain or to present in understandable terms to a human [11].
- **Local fidelity**. Global interpretability implies describing the patterns present in the overall model, while local interpretability describes the reasons for a specific decision on a unique sample. For interpreting a specific observation, we assume it is sufficient to understand how it behaves locally.
- **Model-agnostic**. The goal is to provide a set of techniques that can be applied to any classifier or regressor in contrast to other domain-specific techniques [33].

In practice, LIME creates interpretable explanations for an individual sample by fitting a linear model to a set of perturbed variations of the sample and the resulting predictions as output from a complex-model.

3.5 Predictive Models

The selection of which machine learning model to use highly depends on the problem nature, constraints and limitations that are trying to be solved. In this work, algorithms from different families of machine learning were investigated. Specifically, the Naive Bayes Classifier (NB) and Generalised linear Model (GLM) were investigated as linear models, Random Forests (RF), Gradient Boosting Machines (GBMs) were used to evaluate Decision Tree based ensembles and fully connected Neural Networks (NN) were also assessed. Furthermore, the model ensembling technique of Stacking (STK) was also assessed. Stacking comprises of learning a linear model to predict the target variable based on the output probabilities of multiple machine learning algorithms as features.

3.6 Hotel Model

The first step is to train a machine learning model on individual hotels, as shown is Figure 3. The features used for training this model are not exclusively related to hotels, but also with the session and flight context. Evaluating this model, we get the probability that a certain hotel will be booked for a given location. The model is learned by framing the problem as a supervised classification problem, using the conversion (i.e. click) as a label. Note that for the hotel model, the probabilities of conversion are independent of other hotels presented in the session. This leads to several advantages:

- Cold start problem: the model does not penalise items or users that have not been recommended yet, since no hotel identifier or personally identifiable information is used. [31].
- Dimensionality reduction: the output probabilities of the hotel model can be interpreted as a feature that comprises the relationship between the independent variables and the target variable. This is a key concept of the Cascade Generalization technique, thus the output of the hotel model is combined with the features to create the feature vector for the session model, as shown in 4.

Note that the features used as input to the hotel model are discussed in Section 4.

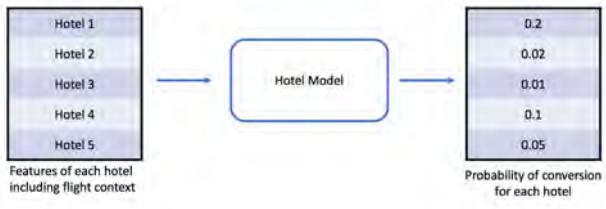

Figure 3: Sketch of the Hotel Model. The machine learning model is trained to predict the probability that each hotel will be booked.

3.7 Session Model

The second machine learning model predicts whether a session leads to a conversion or not, see Figure 4. A session is composed of five different hotels and the aim of the recommender system is to propose a set of hotels that results in the user booking any one of them. Aggregates of the features from the Hotel Model (contextual, passenger, and hotel features) are used, as well as the hotel probabilities obtained from the hotel model. The numerical features related with the hotels are aggregated in different ways (max, min, std and avg of price and probability for example). The features related with the context do not change (e.g. attributes about the session or the flight) as these are identical for each element in the session.

Figure 4: Sketch of the session model pipeline. This machine learning model predicts the probability that a session leads to a conversion, given a list of hotels. This is achieved using cascaded machine learning in which the hotel model predictions are used as features to the session model.

3.8 Session Builder

The Session Model estimates the conversion probability of the session using contextual and content information. Thus, part of the session builder is to create and evaluate new lists of hotels to determine whether these lists will result in higher conversion probability than the original list. Figure 5 shows how this process is performed. First, a reference session with the recommendations, given by an existing rule based system, is scored. For each of the proposed hotels, we estimate the booking probability of each hotel using the Hotel Model. Next, we can calculate the booking probability at session level, using the probabilities of the Hotel Model as an input feature of the Session Model. Then, we aim to improve the conversion probability of the session by removing one of the hotels from the list and introducing a new one. After including the new hotel, if the booking probability of the current session is greater than the probability of the previous session, then this new hotel list is the one that will be proposed to the user.

A rule must be defined to select the hotel to remove and which new hotel to introduce in the recommendation list. Once we have trained the Session Model, we can analyse the feature importance of the variables for the positive cases that were correctly classified (i.e. true positive cases). With the Local Interpretable Model-Agnostic Explanations model [29], we can understand the behaviour of the model for these particular instances. Based on the importance of features from LIME, a heuristic can be defined to replace a hotel from the list in order to improve the session conversion probability.

Note that the LIME analysis is performed only on true positive cases from the training set. In this dataset, the classes are highly imbalanced due to a low conversion rate, as such standard feature analysis techniques may be overly influenced by negative samples, i.e., sessions which did not result in clicks. As LIME is designed to be used on individual decisions, a linear model is fitted and analysed for each true positive. The feature weights for each linear model are then averaged, given a feature importance ranking for all correctly classified converted sessions.

3.9 Evaluation Metrics

As with many conversion problems, the classes are highly imbalanced, and as such the metrics used to assess performance must be carefully chosen.

F-measure (F_β). The generalization of the F_1 metric is given by [7]:

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R}$$

β is a parameter that controls a balance between precision P and recall R. When $\beta = 1$, F_1 comes to be equivalent to the harmonic mean of P and R. If $\beta > 1$, F becomes more recall-oriented (by placing more emphasis on false negatives) and if $\beta < 1$, it becomes more precision oriented (by attenuating the influence of false negatives). Common used metrics are the F_2 and $F_{0.5}$ scores.

Area Under the ROC curve. The receiver operating characteristic (ROC) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold levels. However, this can present an optimistic view of a classifier performance if there is a large skew in the class distribution because the metric takes into account true negatives.

Average Precision (AP). The precision-recall curve is a similar evaluation measure that is based on recall and precision at different threshold levels. An equivalent metric is the Average Precision (AP) which is the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold as the weight:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

Precision-recall curves are better for highlighting differences between models for unbalanced datasets due to the fact that they evaluate the fraction of true positives among positive instances. In highly imbalanced settings, the AP curve will likely exhibit larger differences and will be more informative than the area under the ROC curve. Note that the relative ranking of the algorithms does not change since a curve dominates in ROC space if and only if it dominates in PR space [10, 30].

4 DATA

4.1 Hotel Recommendation Logs

The dataset in this study consists of 715,952 elements. Out of these recommendations, there are a total of 3,588 clicks, which are considered conversions. Therefore, the dataset is unbalanced since only 0.5% of the instances are session conversions.

Each row contains information regarding the context of the session, the recommended hotel, and whether the recommendation led to a conversion. In particular, the features are the number of recommendations (from 1 to 5), date of the recommendation, country where the booking was made, country where the passenger is traveling, hotel identifier, hotel provider identifier, price of the hotel at time of the recommendation, price currency and whether the recommendation led to a conversion. Additionally, the logs were enriched with supplementary information regarding each hotel including a hotel numerical rating (from 0 to 5), hotel categorical rating and the hotel chain.

4.2 Passenger Name Record

In the travel industry, a Passenger Name Record (PNR) is the basic form of computerized travel record. A PNR is a set of data created when a travel reservation is made. PNRs include the travel itinerary

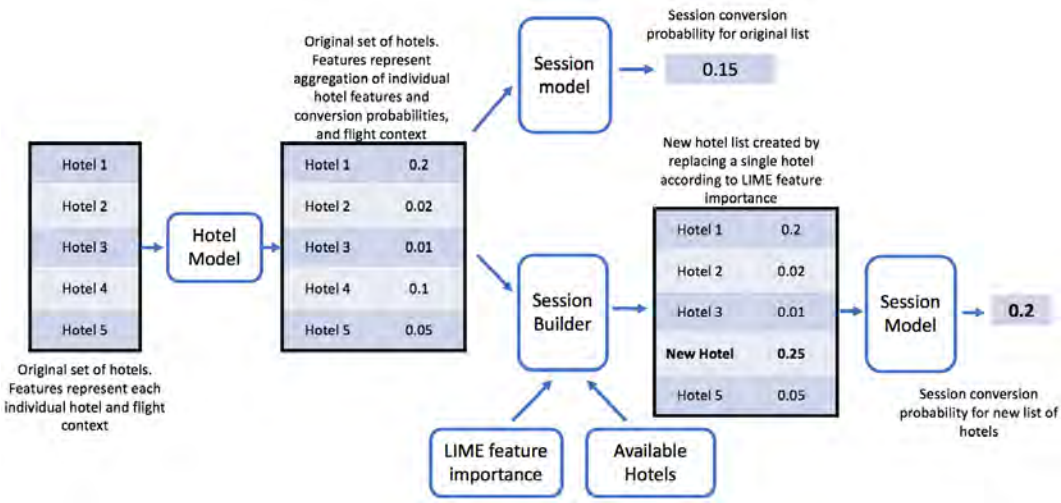

Figure 5: Sketch of the full recommendation pipeline. The session builder is designed to select hotels which will maximise the session conversion, based on the LIME feature importance of the session model.

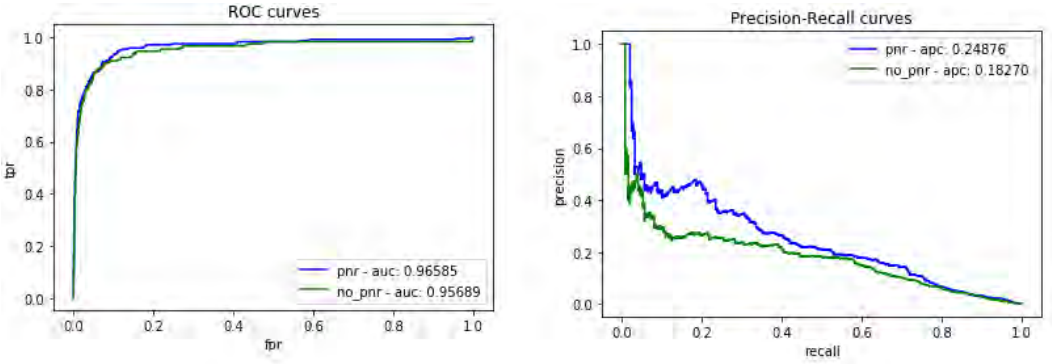

Figure 6: Representation of ROC and AP curves for two Random Forest models predicting individual hotel conversion with and without the PNR data.

information (e.g., flights number, dates) and the passenger information (e.g., name, gender, and somethime passport details). A PNR may also include many other data elements such as payment information (currency, total price, etc), additional ancillary services sold with the ticket (such as extra baggage and hotel reservation) and other airline related information (cabin code, special meal request, etc).

For the purpose of this study, we retrieve and extract features related with the air travel of the traveller. These include the date of PNR creation, airline code, origin city, destination city, date of departure, time of departure, date of arrival, time of arrival, days between the departure and booking date, travel class, number of stops (if any), duration of the flight in minutes (including stops) and the number of days the passenger is staying at the destination.

5 RESULTS

Table 1 shows the results of the experiment comparing different algorithms for the hotel model in terms of AUC, AP, F_1 and $F_{0.5}$ scores. In Figure 6, the ROC and AP curves can be seen in detail. The low AUC value for the GLM model and Naive Bayes Classifier suggest that linear classification techniques do not lead to the best results and more complex models are needed to correctly represent the data. The non-linear techniques have closer results, with the Random Forest obtaining the best values for AP, F_1 and $F_{0.5}$. A Stacked Ensemble using all the previous models is created but it does not improve the previous outcome.

5.1 Contribution of PNR data

The PNR data is an important attribute since it contains rich attributes related to the trip and passenger. However, is this case personally identifiable information is not used in the recommender system, thus the PNR features help to provide context about the

Table 1: Summary of AUC, AP, F_1 and $F_{0.5}$ metrics for the hotel model.

Model	AUC	AP	F1	F0.5
GLM	0.625	0.128	0.247	0.274
NBC	0.819	0.058	0.175	0.159
RF	0.966	**0.249**	**0.320**	**0.334**
GBM	0.953	0.210	0.294	0.288
NN	0.965	0.165	0.245	0.219
STK (all)	0.924	0.182	0.271	0.288
STK (RF + NN)	**0.969**	0.242	0.314	0.284

trip rather than the traveller. Incorporating this data to the models substantially enhanced their performance, as can be observed in Figure 6. Features of the PNR including the number of travellers in the booking and trip duration, among others, contributed to an increase in area under the PR curve from 0.183 to 0.249.

5.2 Session Model

After we have trained the hotel model, we predict individually the probability of conversion of a hotel. Then, we create the sessions based on 5 recommended hotels.

In Table 2 the results are shown. In this case, the best model for both AUC and AP is the Stacked Ensemble composed of a Random Forest, a Generalized Linear Model and a Naïve Bayes Classifier. Although the $F_{0.5}$ score of the GBM model is slightly better than the STK model, the latter clearly outperforms the rest of the metrics.

5.3 Feature Importance

After the Session model has been trained, we analyse its feature importance to study which variables contribute the most to the model using LIME. Concretely, we evaluate the model on the true positive instances from the training dataset, since we want to optimise the conversion.

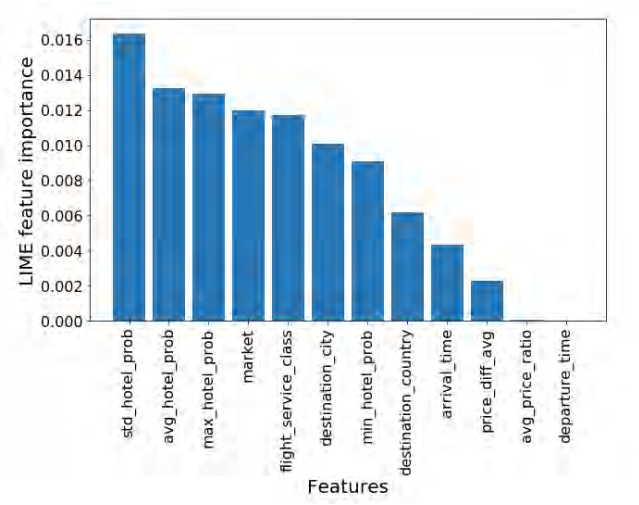

Figure 7: Feature importance of the true positive cases from the Session Model using LIME.

Table 2: Summary of AUC, AP, F_1 and $F_{0.5}$ metrics for the session model.

Model	AUC	AP	F1	F0.5
GLM	0.822	0.395	0.520	0.538
NBC	0.933	0.342	0.467	0.408
RF	0.971	0.446	0.529	0.508
GBM	0.958	0.383	0.531	**0.542**
NN	0.967	0.433	0.483	0.467
STK (RF + GLM + NBC)	**0.972**	**0.453**	**0.539**	0.529

As can be seen in Figure 7, the most important features according to LIME are all derived from the hotel model: the standard deviation, maximum, and average individual hotel conversion probabilities. Some features which are important to the model such as "market" (country where the booking is made from), the flight class of service, the destination city, and arrival and departure times of the flight can not be used to manipulate the results of the session builder, as these are all part of context of the recommendation. Features extracted from prices (the difference between the average price and the minimum, and the ratio of the lowest price to the average price) are also considered important by the LIME model, but rank lower than many hotel conversion probability features.

As the standard deviation of the individual hotel conversions is the most important criteria, the following rule for the session builder is defined: from the original hotel list remove one hotel with the closest conversion probability to the mean conversion probability of the list, and replace it with the hotel with the highest conversion probability from the set of available hotels for a particular city.

5.4 Simulated conversion using Hotel List Builder

Results from the hotel list builder are shown in Table 3 for the two largest cities in the dataset and for the complete dataset. For both cities, we observe a large increase in conversion when using the LIME based session builder. However, a brute force approach to evaluating all possible lists does lead to higher conversion rates, at the cost of a significant increase in processing time. When we consider the complete dataset, we once again observe a large increase in conversion from the baseline for the LIME model. With respect to brute force, we observe that the LIME session builder performs much closer to the brute force builder in terms of conversion. This is attributed to the impact of smaller cities in the complete dataset, and thus less choice in hotels for the builders, resulting in the LIME session builder finding the optimal list. Additionally, on the complete dataset, the processing time of the brute force builder is 2.8 times the duration of the LIME builder, whereas larger gains were observed on the individual cities, where more options for hotels were available.

6 DISCUSSION

In this study, an algorithm was created to improve hotel recommendations based on historical hotel bookings and flight booking attributes. Different machine learning models are used in a cascaded

Table 3: Conversion rates and processing times for two large cities and the complete dataset. The baseline performance is given prior to any optimisation of the hotel lists, the LIME based optimisation is compared to brute force.

	Nice	Barcelona	Complete
Base Conversion	0.0019	0	0.0005
Conversion LIME	0.0207	0.0089	0.0019
Conversion brute	0.0338	0.0125	0.0026
Processing time LIME	23s	23s	4h48m
Processing time brute	314s	496s	13h36m

fashion. First, a model estimates the conversion probability of the individual hotels independently. Note that adding trip context, via PNR based features, resulted in better PR AUC. The output of the first model is then combined with aggregates of the hotels in the list in order to create a feature vector for the session model to estimate the conversion probability that any hotel in the list will be converted. LIME analysis revealed that the hotel model conversion probabilities are the most important features, specifically the standard deviation, mean and maximum individual hotel conversion probabilities in the list. This allows for a simple heuristic to be defined to increase the session conversion probability. In this study, a single change is performed in the list of hotels, however this could be expanded to allow multiple changes.

Variations on this pipeline could also be considered, for instance LIME is used in this study for feature importance ranking in the session builder, however recently a similar methodology was proposed using a mixture regression model referred to as LEMNA [14].

Here, the session builder relies on insights gained from analysis of the feature importance ranking of the session model using LIME over all sessions which lead to a conversion. Thus, the same heuristic is applied to all datapoints in the session builder. However, a key aspect of LIME is that it provides an interpretation of a model for a single datapoint. As such, an evolution of the approach would be to compute the most important features for each recommendation in real time, and to use the information to build an optimal hotel list based on the attributes most likely to lead to conversion.

REFERENCES

[1] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research* 11, Jun (2010), 1803–1831.

[2] Eric Bauer and Ron Kohavi. 1998. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning* 36, 1 (1998), 2.

[3] Yolanda Blanco-Fernandez, Jose J Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, and Martin Lopez-Nores. 2008. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics* 54, 2 (2008).

[4] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (July 2013), 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[5] Robin Burke and Maryam Ramezani. 2011. Matching recommendation technologies and domains. In *Recommender systems handbook*. Springer, 367–386.

[6] Marcirio Silveira Chaves, Rodrigo Gomes, and Cristiane Pedron. 2012. Analysing reviews in the Web 2.0: Small and medium hotels in Portugal. *Tourism Management* 33, 5 (2012), 1286–1287.

[7] Nancy Chinchor. 1992. MUC-4 Evaluation Metrics. In *Proceedings of the 4th Conference on Message Understanding (MUC4 '92)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 22–29. https://doi.org/10.3115/1072064.1072067

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[9] Antónia Correia, Patricia Oom do Valle, and Cláudia Moço. 2007. Why people travel to exotic places. *International Journal of Culture, Tourism and Hospitality Research* 1, 1 (2007), 45–61.

[10] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 233–240.

[11] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. (2017).

[12] Expedia. 2013. Retail and Travel Site Visitation Aligns As Consumers Plan and Book Vacation Packages. https://advertising.expedia.com/about/press-releases/retail-and-travel-site-visitation-aligns-consumers-plan-and-book-vacation-packages

[13] João Gama and Pavel Brazdil. 2000. Cascade generalization. *Machine Learning* 41, 3 (2000), 315–343.

[14] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 364–379.

[15] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 241–250.

[16] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[17] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction* 27, 3-5 (2017), 351–392.

[18] Ingrid Jeacle and Chris Carter. 2011. In TripAdvisor we trust: Rankings, calculative regimes and abstract systems. *Accounting, Organizations and Society* 36, 4 (2011), 293–309.

[19] Michael Kenteris, Damianos Gavalas, and Aristides Mpitziopoulos. 2010. A mobile tourism recommender system. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 840–845.

[20] Dae-Young Kim, Yeong-Hyeon Hwang, and Daniel R Fesenmaier. 2005. Modeling tourism advertising effectiveness. *Journal of Travel Research* 44, 1 (2005), 42–49.

[21] Ron Kohavi, David H Wolpert, et al. 1996. Bias plus variance decomposition for zero-one loss functions. In *ICML*, Vol. 96. 275–83.

[22] Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, Isabelle Guyon, Don Henderson, RE Howard, and W Hubbard. 1989. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine* 27, 11 (1989), 41–46.

[23] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. 2012. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 115–122.

[24] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[25] Stanley Loh, Fabiana Lorenzi, Ramiro Saldaña, and Daniel Licthnow. 2003. A tourism recommender system based on collaboration and text analysis. *Information Technology & Tourism* 6, 3 (2003), 157–165.

[26] Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 195–204.

[27] Julia Neidhardt, Leonhard Seyfang, Rainer Schuster, and Hannes Werthner. 2014. A picture-based approach to recommender systems. *Information Technology & Tourism* 15, 1 (sep 2014), 49–69. https://doi.org/10.1007/s40558-014-0017-5

[28] Andreas Papatheodorou. 2001. Why people travel to different places. *Annals of tourism research* 28, 1 (2001), 164–179.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.

[30] Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one* 10, 3 (2015), e0118432.

[31] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.

[32] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. 2012. Making machine learning models interpretable.. In *ESANN*, Vol. 12. Citeseer, 163–172.

[33] Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. 2014. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3566–3573.

Designing a Conversational Travel Recommender System Based on Data-Driven Destination Characterization

Linus W. Dietz
Department of Informatics
Technical University of Munich
Garching, Germany
linus.dietz@tum.de

Saadi Myftija
Department of Informatics
Technical University of Munich
Garching, Germany
saadi.myftija@tum.de

Wolfgang Wörndl
Department of Informatics
Technical University of Munich
Garching, Germany
woerndl@in.tum.de

ABSTRACT

Recommending complex, intangible items in a domain with high consequences, such as destinations for traveling, requires additional care when deriving and confronting the users with recommendations. In order to address these challenges, we developed CityRec, a destination recommender that makes two contributions. The first is a data-driven approach to characterize cities according to the availability of venues and travel-related features, such as the climate and costs of travel. The second is a conversational recommender system with 180 destinations around the globe based on the data-driven characterization, which provides prospective travelers with inspiration for and information about their next trip. An online user study with 104 participants revealed that the proposed system has a significantly higher perceived accuracy compared to the baseline approach, however, at the cost of ease of use.

KEYWORDS

Tourism recommendation, Data mining, Cluster analysis, Conversational recommender systems

1 INTRODUCTION

In complex recommendation domains, such as the recommendation of tourist destinations, tweaking the algorithmic accuracy ad ultimo brings diminishing returns. It has been shown that the embedding of the algorithm in an adequate user interface is of similar importance [16]. Thus, in this paper, we present a data-driven conversational destination recommender system that has two contributions: it presents a novel, data-driven approach for characterizing destinations on user-understandable dimensions and shows how this characterization can be facilitated in a conversational recommender. This approach can be seen as an evolution of Burke's FindMe Approach [3] in the area of tourism. We thoroughly evaluated the system from the users' perspective to understand the effect of critiquing on the perceived accuracy of the recommendations and the satisfaction of the users from using the system.

After the literature review in the subsequent section, we will present the proposed method for characterizing destinations to realize content-based recommendations. Section 4, presents the the design and evaluation of the conversational recommender system that heavily relies on the previous characterization. We conclude our findings and point out future work in Section 5.

2 RELATED WORK

Tourism recommendation is inherently complex and has several facets. Borràs et al. enumerate four general functionalities of tourism recommender systems [2]: recommend travel destinations and tourist packs [17, 31], suggesting attractions [18], trip planners [10, 12], and social aspects [13]. In this paper, we focus on the first aspect and acknowledge that there are further definitions [1]. Herein, "destination" refers to cities. The challenge in recommending cities to a user at home arises from the intangibility of the items and the high emotional involvement [33]. It has been shown that leisure travel has a positive effect on an individual's happiness; however, it does not impact the overall life satisfaction, which has been attributed to poor tourism products [23]. An alternative conclusion could be that travelers visit the wrong places. This gives rise to researching improved destination recommender systems that can efficiently and effectively capture the user's preferences to overcome the cold start problem [5]. Given the characteristics of this domain, Burke and Ramezani suggested either the content-based [27] or the knowledge-based [3] paradigm [7].

In traditional information retrieval or static content-based recommendation, continuously querying for relevant items does not necessarily lead to better results [4]. Instead, a directed exploration of the search space using a conversational method is more promising [8, 11]. Burke et al. proposed and evaluated the FindMe approach [6], which allows the critiquing of single items so that the user can refine the recommendations iteratively until she is satisfied with the result. More advanced approaches on this topic are those of McCarthy et al., who propose a method to generate compound critiques [19], and McGinty and Smyth, who use the adaptive selection strategy to ensure diverse, yet fitting recommendations over the course of several critiquing cycles [21]. Recently, Xie et al. showed that incorporating the user experience into a critiquing system can improve the performance and recommendations at a reduced effort by the user [35]. In this study, we present a recommender system leveraging the potentials of the interplay between data science and user interface design. The items are characterized by a multidimensional space of features, which are intuitively understandable by the user and can then be critiqued in any direction. To overcome the problem of skeptical users hesitating to reveal their complete preferences [29] and the observation that users find it difficult to assess their exact preferences until when they are dealing with the actual set of offered options [26], the proposed method uses a mixture of explicit preference elicitation methods.

Using the content-based recommendation paradigm, one has to choose a domain model and distance metric to compute the most fitting items for the user. Such models can be realized through ontologies as done in SigTur [22] or in a the work of Grün et al. [14]. The latter is an example of ontologies being used to refine user profiles by enriching the generic preferences of a tourist through more specific interests. More often, items are simply characterized

Table 1: Raw values of exemplary cities

City	Venues	Arts	Food	Nightlife	Outdoors	Cost Index	Temperature	Precipitation
Rome	36,848	1,995	12,264	2,063	3,482	69.03	15.7°C	798mm
Mexico City	213,612	12,158	83,225	16,780	19,330	34.18	15.9°C	625mm
Cologne	16,163	966	4,107	1,144	2,127	67.36	10.1°C	774mm
Penang	50,647	2,193	21,389	1,686	5,273	43.98	25.7°C	1,329mm
Cordoba	3,636	246	1,282	427	379	55.11	17.8°C	612mm

using a multidimensional vector space model. In this case, the challenge is how to assign each item a value on each dimension, which is commonly done using expert knowledge. For instance, Herzog and Wörndl [15, 34] characterized regions using travel guides and their own expert knowledge. Neidhardt et al. developed the Seven Factor Model of tourist behavioral roles [24] based on the Big Five Factor Model [20] and a factor analysis of existing tourist roles [36]. Although they showed its merit in subsequent publications [25], a common drawback with approaches based on expert judgment is their scalability to large quantities of items and the dependency on the accuracy of human judgment. To overcome this, they proposed a strategy [32] for characterizing destinations within the Seven Factor Model. Using a huge data set of 16,950 destinations annotated with 26 motivational ratings and 12 geographical attributes, they proposed two competing methods, cluster analysis and regression analysis, to map the destinations to the vector space of the Seven Factor Model. In terms of destination characterization, this approach is the most similar to the one we proposed. The main difference is that our data model is directly defined via the data from the destinations and we are not dependent on expert ratings, which is an advantage when scaling the approach [9].

3 DESTINATION CHARACTERIZATION

The characterization of destinations such as regions or cities is a challenging task. What are the characteristics of a city for tourists to base their decision on whether to visit it or not? Previous approaches have relied on expert assessment [15, 32], but the shortcomings are a potential lack of objectivity and scalability as it is quite costly to rate myriads of destinations around the world. Thus, we propose a data-driven approach to characterize cities on the basis of the variety of venues per category. The underlying assumption is that, in a city with many restaurants, the travelers have plenty of options; thus, the quality of experience in the food category is high. Conversely, a city with very few cultural sites will be less interesting to a traveler that is interest in this topic. This section discusses how we collected data about venues and aggregated them to determine the touristic value of each city.

3.1 Collecting Venue Information

There are several providers of information about destinations. After performing a comparison of providers, such as Google Maps, Facebook Places, Yelp, OpenStreetMap, and some others, we decided to use the Foursquare Venue API[1], as it offers sufficient rate limitations and allows us to specify coordinates of a bounding box in the request parameters. The deciding argument for Foursquare was the detailed categorization of venues from its taxonomy[2].

[1]https://developer.foursquare.com/docs/api/venues/search
[2]https://developer.foursquare.com/docs/resources/categories

3.2 Characterizing Cities Based on Venue Data

We collected a data set of 5,723,169 venues in 180 cities around the world. Foursquare organizes its venues in a tree of 10 top-level categories, however, we only analyzed the ones relevant for characterizing the cities for travelers: *Arts & Entertainment*, *Food*, *Nightlife*, and *Outdoors & Recreation*. We intend to conceptualize these features as a multidimensional vector space model and represent each city as a point in this space. The characterization should approximate the expected experience that a tourist will have at a city.

To determine a city's score for a feature, we analyzed the distribution of the venue categories. Using the distribution instead of the absolute number of venues per category, we eliminated the effect of city size on the category features. Thus, we obtained the ratio of each feature in the city's category distribution by dividing the number of venues per each top level category by the total number of venues in that city. The underlying assumption is that these percentages are indicators of the association level of the city with the feature. This requires the cities to be of at least a certain size as the distribution of small cities is less reliable. Thus, the smallest city considered had at least 1,000 venues, with the median being 7,137. We did not analyze the quality of the venues, i.e., through ratings, as we expected having differences in the assessment of the quality owing to cultural differences.

Characterizing the cities according to their attractions is a first step; however, further features are of the travelers' interest. Using Climate-Data.org[3], we characterized each city using the mean yearly temperature and the mean yearly precipitation. Furthermore, we used Numbeo's "Cost of Living Index"[4], which is a relative cost indicator calculated by combining metrics like consumer goods prices, restaurants, transportation, and so on as an approximate price level of visiting the city. Finally, to account for the city size, we also used the number of venues as a proxy feature for the size of the city. Table 1 shows the raw values of the features.

3.3 Cluster Analysis

To evaluate the characterization of the 180 cities, we performed a cluster analysis, an unsupervised learning method whose goal is to group data items in a way that within the same group, the items are similar to each other, whereas the groups are dissimilar. Because the features of the destinations that we considered have different value ranges, we first applied min-max scaling to give each feature the same weight. To find the best segmentation, we experimented with common clustering algorithms, such as k-means, k-medoids, and hierarchical clustering. To evaluate the quality of the resulting clusters, we looked into metrics like the within-cluster sums of squares and the average silhouette width [30]. The former

[3]https://en.climate-data.org
[4]https://www.numbeo.com/cost-of-living/rankings.jsp

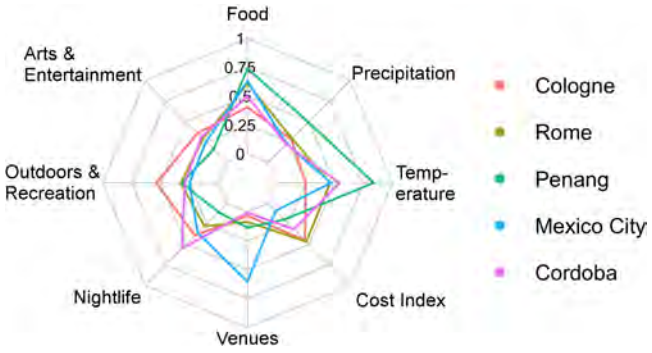

Figure 1: Normalized values of selected destinations

is a measure of the variability of the instances within each cluster, whereas the latter is a measure of how well the instances fit into their assigned cluster, as opposed to all the other clusters.

Using a systematic approach, we obtained the best results using hierarchical clustering and five clusters. The clusters named after the city closest to the centroid are "Cologne, Germany," with 74 Central European and North American cities; "Rome, Italy" with 35 cities in the Mediterranean and Oceania; "Penang, Malaysia" with 48 destinations residing mostly in Asia; "Mexico City, Mexico" with five metropoleis all around the world; and "Cordoba, Spain," with 18 small and relatively warm cities in different continents. Figure 1 shows the normalized values of the five characteristic cities.

4 A DATA-DRIVEN CONVERSATIONAL DESTINATION RECOMMENDER SYSTEM

Having characterized the destinations on eight dimensions, we facilitate it in a content-based critiquing recommender system. CityRec is implemented as a web application using NodeJS[5] and ReactJS[6] in the frontend. The codebase comprises about 3,500 lines of code and is available on Github[7]. A demo can be viewed at http://cityrec.cm.in.tum.de.

4.1 User Interaction with CityRec

The recommender system has three steps: (1) initial preference elicitation, shown in Figure 2 (a); (2) refinement through critiquing, shown in Figure 2 (b); and (3) a results page. In Step (1), we obtain the initial scores for the user profile by asking the user to select the destinations that best reflect her preferences from a set of 12 cities. We then construct an initial user model by averaging the feature values of the selected cities. This initial seed of 12 destinations is not random, but a diverse representation of the data set. We fill in the first nine slots by selecting two cities from each of the five previously established destination clusters (one in the case of the small "Mexico City" cluster). The remaining three slots are randomly selected cities to account for the size differences of the clusters. Using this approach, we can generate numerous, diverse, but equivalent shortlists because each cluster is represented. From these 12 cities, the users may choose three to five that best reflect their preferences. If a user does not recognize many cities, she can request another set of cities. Furthermore, a tooltip encourages the user to select cities that she finds generally interesting, including those she has already visited. This ensures that the system has enough data to work with for generating the initial user profile but avoids cases where users select many displayed cities, which end up in generic profiles with averaged-out feature values. The result of this step is an initial profile of the user that resides in the same vector space as the items.

In Step (2), we display a set of four initial destinations, computed using the Euclidean Distance. To give the users more control over their preference profile, we ask them to provide feedback on the initial recommendations by critiquing the cities' features one after another on a five-point Likert Scale: *"much lower" – "lower" — "just right" — "higher" — "much higher."* As can be seen in Figure 2 (b), the user now has more information about the cities, which establishes transparency and enables her to more informed decisions compared to in the first step. Using this feedback, we statically update the user profile scores by -0.2, -0.1, 0, 0.1, or 0.2 to attain a more refined preference model for the user.

Finally, in the last step, Step (3), the user is presented with a results page that shows a ranked list of the top five recommendations and their attributes, which can be explored. This page also contains the questionnaire for the evaluation.

4.2 Experimental Setup

The independent variable of the experiment is the version of the recommender system. Because we wanted to investigate the potential advantages and drawbacks of using critiquing in this domain, we created a baseline system in addition to the previously described critiquing-based recommender. The only difference in the baseline system was that the critiquing step, Step (2), is entirely skipped; that is, the outcome of the initial preference elicitation of Step (1) is the final result and is displayed in the same way as in Step (3).

The dependent variables are the usage metrics, such as the choices made at each step, the time taken to specify the preferences, and the number of clicks. Furthermore, we asked the user to fill out a subset of the ResQue Questionnaire, a validated, user-centric evaluation framework for recommender systems [28].

(Q1) The travel destinations recommended to me by CityRec matched my interests
(Q2) The recommender system helped me discover new travel destinations
(Q3) I understood why the travel destinations were recommended to me
(Q4) I found it easy to tell the system what my preferences are
(Q5) I found it easy to modify my taste profile in this recommender system
(Q6) The layout and labels of the recommender interface are adequate
(Q7) Overall, I am satisfied with this recommender system
(Q8) I would use this recommender system again, when looking for travel destinations

4.3 Results

A total of 104 individuals participated in the online survey from December 2018 to March 2019. Participants (44% females, 56% males)

[5]https://nodejs.org/en/
[6]https://reactjs.org/
[7]https://github.com/divino5/cityrec-prototype

Figure 2 (a): Selection of favorable cities, Step (1)

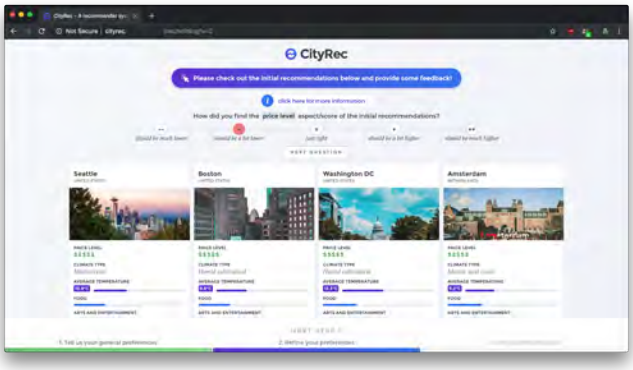

Figure 2 (b): Critiquing of initial recommendations, Step (2)

were recruited by sharing the user study on social media and among groups of friends and colleagues. The self-reported ages were 0–20 (7%), 21–30 (69%,) 31–40 (9%), and 41–50 (5%). Random assignment of the systems was performed after a landing page and had almost equal (51% versus 49%) completion of the survey.

Table 2: Differences between the two systems

Variable	Basel.	Critiqu.	p	W	Sig.
(Q1) Interest match	3.58	**3.88**	**0.043**	645	*
(Q2) Novelty	3.44	3.75	0.118	705	ns
(Q3) Understanding	3.46	3.77	0.073	673.5	ns
(Q4) Tell prefs.	3.73	3.90	0.328	775	ns
(Q5) Modify profile	3.24	3.48	0.17	723.5	ns
(Q6) Interface	**4.15**	3.62	**0.009**	1,044	**
(Q7) Satisfaction	3.66	**3.92**	**0.037**	649	*
(Q8) Future use	3.49	3.67	0.166	724	ns
Time to results	60.92s	**184.07s**	**<0.001**		***
Clicks	6.32	**21.35**	**<0.001**		***
PCC Food	-0.11	-0.01	0.341		ns
PCC Arts	0.05	0.38	0.066		ns
PCC Outdoors	0.02	**0.45**	**0.024**		*
PCC Nightlife	0.2	**0.57**	**0.028**		*

Significance levels: $* p < 0.05$; $** p < 0.01$; $*** p < 0.001$

The upper part of Table 2 shows the differences in the mean values and the significance tests of the dependent variables. The mean values of the ordinal answers to the questionnaire (Q1–Q8) are for viewing purposes only; the test statistic was calculated using the Wilcoxon rank sum test with continuity correction for independent populations. The null hypotheses were that the medians of variables of the two groups are equal. In three cases, (Q1), (Q6), and (Q7), we could refute the null hypothesis, which provides interesting insights into the users' assessment of the system.

In the survey, we also asked the participants to rate their personal importance of tourism-related aspects. Thus, we could compute the Pearson Correlation Coefficient (PCC) between the actual profile from the system and the self-assessment from the survey. The lower part of Table 2 shows these correlations per system and the result of the one-sided Fisher's r-to-Z test for independent samples.

4.4 Discussion

The significant difference in (Q1) shows that the perceived recommendation accuracy is higher, when using the proposed critiquing recommender system, however, at the cost of worse interface adequacy (Q6). This is attributable to the overhead of the critiquing step, Step (2), as it takes triple the time to complete the first two steps and more than triple the number of clicks. Interestingly, the users value higher accuracy more than the adequacy of the interface and the effort as can be seen in the significantly higher user satisfaction (Q7) and the similar levels of potential future use (Q8).

Furthermore, we observed that the user profiles of the critiquing system are significantly higher correlated with the self-assessment in the case of Outdoors & Recreation and Nightlife. This is further evidence that the critiquing recommender version performs better in capturing the preferences of the user. In conclusion, the critiquing version should be preferred as it provides better recommendations from the users' perspective.

5 CONCLUSIONS

In this paper, we proposed an approach for tackling the problem of recommending complex items in the domain of travel recommendation. We characterized destinations around the globe in a user-understandable way and directly used this characterization in an online recommender system. From the evaluation experiments conducted, we discovered an interesting trade-off between the perceived recommendation accuracy and the perceived adequacy of the user interface; however, the users seemed to favor better recommendations over less effort to obtain them.

Because CityRec's source code has been released, it can also serve as a foundation for the community to investigate conversational recommender systems based on data-driven item characterization. The destination characterization showed decent results; however, it would be worthwhile to investigate further useful features of destinations that can be derived from other data sources. In this study, we found that, despite higher perceived accuracy (Q1), the interface adequacy (Q6) was rated lower in the critiquing system. Thus, we regard this study as a first step that is to be extended with a more sophisticated preference elicitation approach using active learning. Furthermore, the behavior of the algorithm, with respect to the diversity of the recommendations, should be analyzed as well.

REFERENCES

[1] David Beirman. 2003. *Restoring Tourism Destinations in Crisis: A Strategic Marketing Approach.* Oxford University Press, Oxford, United Kingdom.

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

[3] Robin D. Burke. 2000. Knowledge-based recommender systems. *Encyclopedia of library and information science* 69, 32 (2000), 180–200.

[4] Robin D. Burke. 2002. Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review* 18, 3 (Dec. 2002), 245–267. https://doi.org/10.1023/A:1020701617138

[5] Robin D. Burke. 2007. Hybrid Web Recommender Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer, Berlin, Heidelberg, 377–408. https://doi.org/10.1007/978-3-540-72079-9_12

[6] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. 1997. The FindMe approach to assisted browsing. *IEEE Expert* 12, 4 (July 1997), 32–40. https://doi.org/10.1109/64.608186

[7] Robin D. Burke and Maryam Ramezani. 2011. *Recommender Systems Handbook.* Springer, Boston, MA, USA, Chapter Matching Recommendation Technologies and Domains, 367–386. https://doi.org/10.1007/978-0-387-85820-3_11

[8] Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction* 22, 1 (April 2012), 125–150. https://doi.org/10.1007/s11257-011-9108-6

[9] Linus W. Dietz. 2018. Data-Driven Destination Recommender Systems. In *26th Conference on User Modeling, Adaptation and Personalization (UMAP '18).* ACM, New York, NY, USA, 257–260. https://doi.org/10.1145/3209219.3213591

[10] Linus W. Dietz and Achim Weimert. 2018. Recommending Crowdsourced Trips on wOndary. In *RecSys Workshop on Recommenders in Tourism (RecTour'18).* Vancouver, BC, Canada, 13–17.

[11] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2016. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review* 20, Supplement C (May 2016), 29–50. https://doi.org/10.1016/j.cosrev.2016.05.002

[12] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. A survey on algorithmic approaches for solving tourist trip design problems. *Heuristics* 20, 3 (June 2014), 291–328. https://doi.org/10.1007/s10732-014-9242-5

[13] Ulrike Gretzel. 2011. Intelligent systems in tourism: A Social Science Perspective. *Annals of Tourism Research* 38, 3 (July 2011), 757–779. https://doi.org/10.1016/j.annals.2011.04.014

[14] Christoph Grün, Julia Neidhardt, and Hannes Werthner. 2017. Ontology-Based Matchmaking to Provide Personalized Recommendations for Tourists. In *Information and Communication Technologies in Tourism*, Roland Schegg and Brigitte Stangl (Eds.). Springer, Cham, 3–16.

[15] Daniel Herzog and Wolfgang Wörndl. 2014. A Travel Recommender System for Combining Multiple Travel Regions to a Composite Trip. In *CBRecSys@RecSys.* Foster City, CA, USA, 42–48.

[16] Joseph A. Konstan and John Riedl. 2012. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction* 22, 1-2 (April 2012), 101–123. https://doi.org/10.1007/s11257-011-9112-x

[17] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized Travel Package Recommendation. In *IEEE 11th International Conference on Data Mining (ICDM '11).* IEEE, Vancouver, BC, Canada, 407–416. https://doi.org/10.1109/icdm.2011.118

[18] David Massimo and Francesco Ricci. 2018. Clustering Users' POIs Visit Trajectories for Next-POI Recommendation. In *Information and Communication Technologies in Tourism*, Juho Pesonen and Julia Neidhardt (Eds.). Springer, Cham, 3–14. https://doi.org/10.1007/978-3-030-05940-8_1

[19] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. 2004. On the dynamic generation of compound critiques in conversational recommender systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems.* Springer, Berlin, Heidelberg, 176–184.

[20] Robert R. McCrae and Oliver P. John. 1992. An Introduction to the Five-Factor Model and its Applications. *Personality* 60, 2 (June 1992), 175–215. https://doi.org/10.1111/j.1467-6494.1992.tb00970.x

[21] Lorraine McGinty and Barry Smyth. 2006. Adaptive Selection: An Analysis of Critiquing and Preference-Based Feedback in Conversational Recommender Systems. *Electronic Commerce* 11, 2 (Dec. 2006), 35–57. https://doi.org/10.2753/jec1086-4415110202

[22] 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

[23] Jeroen Nawijn. 2012. *Leisure Travel and Happiness: An Empirical Study into the Effect of Holiday Trips on Individuals' Subjective Wellbeing.* phdthesis. Erasmus University Rotterdam, Rotterdam.

[24] Julia Neidhardt, Rainer Schuster, Leonhard Seyfang, and Hannes Werthner. 2014. Eliciting the Users' Unknown Preferences. In *8th ACM Conference on Recommender Systems (RecSys '14).* ACM, New York, NY, USA, 309–312. https://doi.org/10.1145/2645710.2645767

[25] Julia Neidhardt, Leonhard Seyfang, Rainer Schuster, and Hannes Werthner. 2015. A picture-based approach to recommender systems. *Information Technology & Tourism* 15, 1 (March 2015), 49–69. https://doi.org/10.1007/s40558-014-0017-5

[26] John W. Payne, James R. Bettman, and Eric J. Johnson. 1993. *The adaptive decision maker.* Cambridge University Press, Cambridge, United Kingdom.

[27] Michael J. Pazzani and Daniel Billsus. 2007. Content-Based Recommendation Systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer, Berlin, Heidelberg, 325–341. https://doi.org/10.1007/978-3-540-72079-9_10

[28] Pearl Pu, Li Chen, and Rong Hu. 2011. A User-centric Evaluation Framework for Recommender Systems. In *Fifth ACM Conference on Recommender Systems (RecSys '11).* ACM, New York, NY, USA, 157–164. https://doi.org/10.1145/2043932.2043962

[29] Francesco Ricci and Quang Nhat Nguyen. 2007. Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. *IEEE Intelligent Systems* 22, 3 (May 2007), 22–29. https://doi.org/10.1109/MIS.2007.43

[30] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics* 20 (Nov. 1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[31] Mete Sertkan, Julia Neidhardt, and Hannes Werthner. 2017. Mapping of Tourism Destinations to Travel Behavioural Patterns. In *Information and Communication Technologies in Tourism*, Brigitte Stangl and Juho Pesonen (Eds.). Springer International Publishing, Cham, 422–434. https://doi.org/10.1007/978-3-319-72923-7_32

[32] Mete Sertkan, Julia Neidhardt, and Hannes Werthner. 2019. What is the "Personality" of a tourism destination? *Information Technology & Tourism* 21, 1 (March 2019), 105–133. https://doi.org/10.1007/s40558-018-0135-6

[33] Hannes Werthner and Francesco Ricci. 2004. E-commerce and Tourism. *Commun. ACM* 47, 12 (Dec. 2004), 101–105. https://doi.org/10.1145/1035134.1035141

[34] Wolfgang Wörndl. 2017. A Web-based Application for Recommending Travel Regions. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization (UMAP '17).* ACM, New York, NY, USA, 105–106. https://doi.org/10.1145/3099023.3099031

[35] Haoran Xie, Debby D. Wang, Yanghui Rao, Tak-Lam Wong, Lau Y. K. Raymond, Li Chen, and Fu Lee Wang. 2018. Incorporating user experience into critiquing-based recommender systems: a collaborative approach based on compound critiquing. *Machine Learning and Cybernetics* 9, 5 (May 2018), 837–852. https://doi.org/10.1007/s13042-016-0611-2

[36] Andrew Yiannakis and Heather Gibson. 1992. Roles tourists play. *Annals of Tourism Research* 19, 2 (Jan. 1992), 287–303. https://doi.org/10.1016/0160-7383(92)90082-z

A Simple Deep Personalized Recommendation System

Pavlos Mitsoulis-Ntompos*
Meisam Hejazinia*
Serena Zhang*
pntompos@expediagroup.com
mnia@expediagroup.com
shuazhang@expediagroup.com
Vrbo, part of Expedia Group

Travis Brady
tbrady@expediagroup.com
Vrbo, part of Expedia Group

ABSTRACT

Recommender systems are critical tools to match listings and travelers in two-sided vacation rental marketplaces. Such systems require high capacity to extract user preferences for items from implicit signals at scale. To learn those preferences, we propose a Simple Deep Personalized Recommendation System to compute travelers' conditional embeddings. Our method combines listing embeddings in a supervised structure to build short-term historical context to personalize recommendations for travelers. Deployed in the production environment, this approach is computationally efficient and scalable, and allows us to capture non-linear dependencies. Our offline evaluation indicates that traveler embeddings created using a Deep Average Network can improve the precision of a downstream conversion prediction model by seven percent, outperforming more complex benchmark methods for online shopping experience personalization.

KEYWORDS

travel, recommender system, deep learning, embeddings, e-commerce

1 INTRODUCTION

Personalizing recommender systems is the cornerstone for two-sided marketplace platforms in the vacation rental sector. Such a system needs to be scalable to serve millions of travelers and listings. On one side, travelers show complex non-linear behavior. For example, during a shopping cycle travelers might collect and weight different signals based on their heterogeneous preferences across various days, by searching either sequentially or simultaneously. Furthermore, the travelers might forget and revisit items in their consideration set [5, 7]. On the other side, marketplace platforms should match each of the travelers with the most personalized listing out of millions of heterogeneous listings. Many of these listings have never been viewed by any traveler or have only been recently onboarded, imposing data sparsity issue. In addition, the context of each trip might be different for travelers within and across different seasons and destinations (e.g. winter trip to mountains with friends, summer trip to the beach with family, etc.). Moreover, such a personalized recommender system should always be available and trained based on the most relevant data, allowing quick test-and-learn iterations, adapting to ever changing requirements of business. This personalized recommender system should suggest handful relevant listings to the millions of travelers visiting site pages (e.g. home page, landing page, or listing detail page), travelers receiving targeted marketing emails, or travelers faced cancelled bookings due to various reasons.

To develop such a recommender system we need to extract travelers' preferences from implicit signals of their interactions using machine learning or statistical-economics models. Given the complexity and scale of this problem, we require high capacity models. While powerful, high-capacity models frequently require prohibitive amounts of computing power and memory, particularly for big data problems. Many approaches have been proposed to learn item embeddings for recommender systems [3, 4, 14, 21], yet learning travelers' preferences from those listing embeddings at scale is still an open problem. Indeed, such a solution needs to capture traveler heterogeneity while being generic and robust to cold start problems. We propose a modular solution that learns listings and traveler embeddings non-linearly using a combination of shallow and deep networks. We used down-funnel booking signals, in addition to implicit signals (such as listing-page-view), to validate our extracted traveler embeddings. We deployed this system in the production environment. We compared our model with three benchmark models, and found that adding these traveler features to the extant feature set in the already-existing Traveler Booking Intent model can add significant marginal values. Our finding suggests that this simple approach can outperform LSTM models, which have significantly higher time complexity. In the next sections we review related work, explain our model, review the results, and conclude.

*Equal contribution to this research.

2 RELATED WORKS

Representation learning has been widely explored for large-scale session-based recommender systems (SBRS), [9, 12, 21], among which collaborative filtering and content-based settings are most commonly used to generate user and item representations [9, 14, 18]. Recent works have addressed the cold start and adaptability problems in factorization machine and latent factor based approaches [11, 17, 22]. Other works have employed non-linear functions and neural models to learn the complex relationships and interactions over users and items on e-commerce platforms [12, 22]. In particular, word2vec techniques with shallow neural networks [16] from the Natural Language Processing (NLP) community have inspired authors to generate non-linear entity embeddings [9] using historical contextual information. State-of-the-art methods have used attention neural networks to aggregate representations in order to focus on relevant inputs and select the most important portion of the context [6]. Attention has been found effective in assigning weights to user-item interactions within the encoder-decoder and Long Short Term Memory (LSTM) architectures and collaborative filtering framework, capturing both long and short term preferences [8, 12, 20]. Similar to the spirit of our work, recent studies suggested simple neural networks, showing promising results in terms of performance, computational efficiency and scalability [2, 10, 26].

3 ARCHITECTURE AND MODEL

In this section, we will describe our model, which is based on the session based local embedding model. Our model has two modular stages. In the first stage, we train a skip-gram sequence model to capture a local embedding representation for each listing, we then extrapolate latent embeddings for listings subject to the cold start problem. In the second stage, we train a Deep Average Network (DAN) stacked with decoder and encoder layers predicting purchase events to capture a given traveler's embedding or latent preference for listings embedding. We also mention a couple of alternatives we evaluated for traveler embeddings. We denote each listing by x_i, so each traveler session $s_k(t_j)$ is defined as a sequence like $x_1, x_2, ...$ for traveler t_j. We denote booking event conditional on listings recently viewed by the traveler with $b_k(t_j|x_{j1}, x_{j2}, , .., x_{jt})$. Our contribution in this paper is mainly the second stage which we validate using a downstream shopping funnel signal.

Skip-gram Sequence Model

The skip-gram model [16] in our context attempts to predict listings x_i surrounded by listings x_{i-c} and x_{i+c} viewed in a traveler session s_k, based on the premise that traveler's view of listings in the same session signals the similarity of those listings. We use a shallow neural network with one hidden layer with lower dimension for this purpose. The training objective is to find the listing local representation that specifies surrounding most similar manifold. More formally the objective function can be specified by the log probability maximization problem as follows:

$$\frac{1}{S}\sum_{s=1}^{S}\sum_{-c\leq j\leq c, j\neq 0}\log p(x_{i+j}|x_i)$$

where c is the window size representing listing context. The basic skip-gram formulation defines $p(x_{i+j}|x_i)$ using softmax function as follows:

$$p(x_{i+j}|x_i) = \frac{\exp(v_{x_{i+j}}^T v_{x_i})}{\sum_{x=1}^{X}\exp(v_x^T v_{x_i})}$$

where v_x and v_{x_i} are input and output representation vector or neural network weights, and X is the number of listings available on our platform. To simplify the task, we used the sigmoid formula, which makes the model a binary classifier, with negative samples, which we draw randomly from the list of all available listings on our platform. Formally, we use the following formula: $p(x_{i+j}|x_i) = \frac{exp(v_{x_{i+j}}^T v_{x_i})}{1+\exp(v_{x_{i+j}}^T v_{x_i})}$ for positive samples, and the following formula for negative ones: $p(x_{i+j}|x_i) = \frac{1}{1+\exp(v_{x_{i+j}}^T v_{x_i})}$.

We have two more issues to address, sparsity and heterogeneity in views per item. It is not uncommon to observe long tail distribution of views for the listings. For this purpose we leverage approaches mentioned by [16] wherein especially frequent items are downsampled using the inverse square root of the frequency. Additionally, we removed listings with very low frequency. To resolve the cold start issue, we leverage the contextual information that relates destinations (or search terms) to the listings based on the booking information. Formally, considering that the destinations $d_1, d_2, ..., d_D$ are driving $p_{id_1}, ..., p_{id_D}$, proportion of the demand for a given listing, we form the expectation of the latent representation for each location using $v_d = \frac{1}{N}\sum_{l=1}^{L} p_{ld} v_{x_l}$, where N is the normalizing factor and L is the total number of destinations. Then, given latitude and longitude of the cold listing (for which we have no data), we form the belief about the proportion of demand driven from each of the search terms $p_{jd_1}, ..., p_{jd_D}$. Then, we use our destination embedding from the previous step to find the expected listing embedding for the cold listing as follows $v_{x_j} = \sum_{d=1}^{D} p_{jd} v_d$.

Deep Average Network and Alternatives

In the second stage, given the listing's embedding from the previous stage we model traveler embeddings using a

sandwiched encoder-decoder non-linear Relu function. In contrast to relatively weak implicit view signals, in this stage we leverage strong booking signals as a target variable based on historical traveler listing interaction. We have various choices for this purpose including Deep Average Network with Auto-Encoder-Decoder, Long Short Term Memory (LSTM), and Attention Networks. The simplest approach is to take the point-wise average of the embedding vector and use it directly in the model. The second approach could be to feed the average embedding into a dimensionality expansion and reduction non-linear encoder-decoder architecture, or Deep Average Network to extract the signals [10]. The third approach could incorporate LSTM network [13, 19], testing the hypothesis that the traveler signals information that they gathered by looking at different listings in the shopping funnel. The fourth approach could have an attention layer on the top of LSTM [25], hypothesizing that they allocate different weights on various latent features before their booking.

We take a probabilistic approach to model traveler booking events $P(Y_j)$ based on the embedding vectors of historical units they have interacted with $v_{j1}, , .., v_{jt}$. Formally, given the traveler embeddings (or last layer of the traveler booking prediction neural network $f(v_{j.})$), the probability of the booking is defined as:

$$P(Y_j|v_{j1}, v_{j1}, , .., v_{jt}) = \text{sigmoid}(f(v_{j.})) \tag{1}$$

where, the Deep Average Network layers and f are defined as:

$$f(v_{j.}) = \text{relu}(\omega_1 \cdot h_2(v_{j.}) + \beta_1) \tag{2}$$

$$h_1(v_{j.}) = \text{relu}(\omega_2 \cdot h_1(v_{j.}) + \beta_2) \tag{3}$$

$$h_2(v_{j.}) = \text{relu}(\omega_3 \cdot \frac{1}{k} \sum_{i=1}^{t} v_{ji}) + \beta_3) \tag{4}$$

Alternatively, we can use an LSTM network with forget, input, and output gates as follows:

$$f(v_j^t) = \text{sigmoid}(\omega_f[h_t, v_j^t] + \beta_f) \cdot f(v_{j.}^{t-1}) + \text{sigmoid}(\omega_i[h_t, v_j^t] + \beta_i) \cdot \tanh(\omega_c[h_{t-1}, v_j^t] + \beta_c) \tag{5}$$

And finally, we can also use an attention network on the top of LSTM network as follows:

$$f(v_j) = \text{softmax}(\omega^T \cdot h_T)\tanh(h_T) \tag{6}$$

where $\omega_., \beta_.$ are weight and bias parameters to estimate and h_t represents the hidden layer parameter or function to estimate.

Among these models, DAN is more consistent with Occam's razor principle, so it is more parsimonious, and faster to train. However, LSTM and Attention Networks on the top of it are more theoretically appealing. As a result, from the pragmatic stand point, for millions of listings and travelers DAN seems to be more appealing for deployment as depicted in Figure 1.

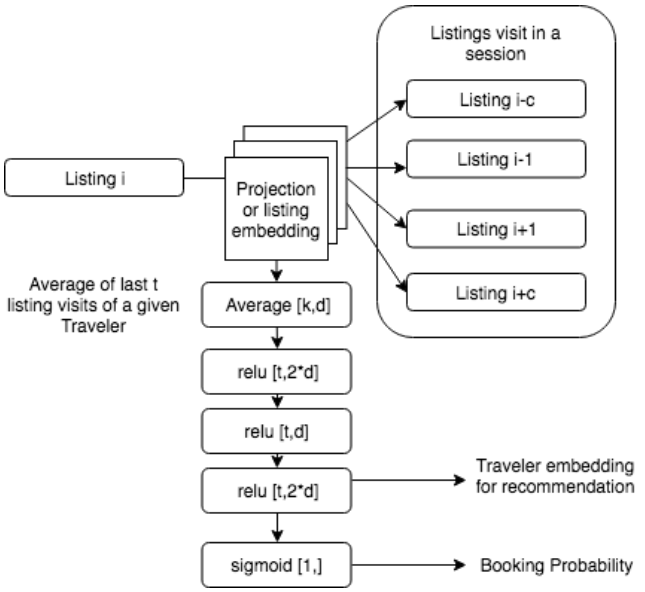

Figure 1: Deep Average Network (DAN) on the top of skip-gram network.

We use adaptive stochastic gradient descent method to train the binary cross entropy of these neural networks. The last question to answer is how are we planning to combine the traveler and listing embedding for personalized recommendations. This is a particularly challenging task as traveler embeddings is non-linear projection of listings embedding with a different dimension. As a result, they are not in the same space to compute cosine similarity. We have various choices for this solution, including approaches such as factorization machine and svm with kernel that allow modeling higher level interactions at scale. We defer the study of this approach to our next study.

4 EXPERIMENTS AND RESULTS

In this section we describe the experimental setup, and the results obtained when comparing the accuracy uplift of our Deep Average Network based approach to various baselines on a downstream conversion prediction model. The Traveler Booking Intent XGBoost model is such a downstream model. It is trained using LightGBM [15] and uses a rich set of hand-crafted historical session-based product interaction features in order to predict the booking intent probability[1]. In order to evaluate offline our proposed methodology, we

[1]We call it booking intent as our model predicts booking request from travelers, which needs a couple of steps to be confirmed as booking.

concatenated the hand-crafted features with the traveler embeddings, generated by all different model settings.

The three baseline methods that we compare against our proposed Deep Average Network on the top of Skip-Gram include the following:

(1) **Random**: a heuristic rule that chooses a random listing embedding, among those listings a traveler has previously interacted with, in the current session.
(2) **Averaging Embeddings**: a simple point-wise averaging of listing embeddings a traveler has previously interacted with, in the current session.
(3) **LSTM with Attention**: A recurrent neural network, inspired by [13, 19, 23], that uses LSTM units and an attention mechanism on top of it in order to combine embeddings of listings a user has previously interacted with, in the current session.

Datasets

For the experiments, anonymized clickstream data is collected for millions of users from two different seven-day periods. Specifically, the click stream data includes user views and clicks of listing detail page logs, search requests, responses, views and clicks logs, homepage views and landing page logs, conversion events logs, per visitor and session. The first click-stream dataset was used to generate embeddings using Deep Average Network and the LSTM with Attention. The second click-stream dataset was used to evaluate the learned embeddings on the Traveler Booking Intent Model. We split each of the data sets into train and test set by 70:30 proportion randomly, based on users. In other words, users that are in the train set are excluded from the test set, and vice versa.

Results

We ran our training pipeline on both CPU and GPU production systems using Tensorflow [1]. We cleaned up the data using Apache Spark [24], and the input data to training pipeline had observations from millions of traveler sessions. The training process for LSTM models typically took 3 full days of time, while training DAN took less than 8 hours on CPU. Given that our recommender system needs to be iterated fast for improvement and infer in real-time with high coverage, DAN model scales better. Moreover, we modified the cost function to give more weight to minority class (i.e. positive booking intent) in order to combat the imbalanced classes in the data sets.

We evaluated the performance of the Traveler Booking Intent model on the different settings using the test data set based on AUC, Precision, Recall and F1 scores. The best results of each model are shown in Table 1. It shows that our proposed Deep Average Network approach contributes more uplift to the downstream Traveler Booking Intent model.

Table 1: Comparison between Model Settings

	Performance Metrics			
Algorithm	AUC	Precision	Recall	F-Score
Random	0.973	0.821	**0.633**	0.715
Averaging Embeddings	0.971	0.816	0.628	0.71
LSTM + Attention	0.976	0.877	0.62	0.727
DAN	**0.978**	**0.888**	0.628	**0.735**

Moreover, Table 2 shows the performance improvement to the Traveler Booking Intent (TBI) model when the Deep Average Network generated traveler embeddings are concatenated to the initial hand-crafted features.

Table 2: Performance Uplift to TBI Model

	Performance Metrics			
Settings	AUC	Precision	Recall	F-Score
Only Hand-Crafted Feat.	0.975	0.817	**0.651**	0.724
Hand-Crafted + DAN Feat.	**0.978**	**0.888**	0.628	**0.735**

We noticed that the Deep Average Network traveler embeddings have competitive predictive power compared to the hand-crafted ones in the downstream TBI model. Based on random re-sampling the dataset and re-running the pipeline, we find that our results are reproducible.

5 CONCLUSION

We presented a method that combines deep and shallow neural networks to learn traveler and listing embeddings for a large online two-sided vacation rental marketplace platform. We deployed this system in the production environment. Our results show Deep Average Networks can outperform more complex neural networks in this context. There are various avenues to extend our study. First, we plan to test attention network without LSTM. Second, we plan to infuse other contextual information into our model. Third, we want to build a scoring layer that combines traveler and listing embeddings to personalize recommendations. Finally, we plan to evaluate numerous spatio-temporal features, representational learning approaches, and bidirectional recurrent neural networks in our framework.

6 ACKNOWLEDGMENTS

This project is a collaborative effort between the recommendation, marketing data science and growth marketing teams. The authors would like to thank Ali Miraftab, Ravi Divvela, Chandri Krishnan and Wenjun Ke for their contribution to this paper.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/

[2] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. (2016).

[3] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks.. In *RecTemp@ RecSys*. 57–59.

[4] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. 2018. Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 352–356.

[5] Hector Chade, Jan Eeckhout, and Lones Smith. 2017. Sorting through search and matching models in economics. *Journal of Economic Literature* 55, 2 (2017), 493–544.

[6] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. 2019. An Attentive Survey of Attention Models. *arXiv preprint arXiv:1904.02874* (2019).

[7] Babur De los Santos, Ali Hortaçsu, and Matthijs R Wildenbeest. 2012. Testing models of consumer search using data on web browsing and purchasing behavior. *American Economic Review* 102, 6 (2012), 2955–80.

[8] Simen Eide and Ning Zhou. 2018. Deep neural network marketplace recommenders in online experiments. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 387–391.

[9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1809–1818.

[10] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 1681–1691.

[11] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems* 27 (2014).

[12] Thom Lake, Sinead A Williamson, Alexander T Hawk, Christopher C Johnson, and Benjamin P Wing. 2019. Large-scale Collaborative Filtering with Product Embeddings. *arXiv preprint arXiv:1901.04321* (2019).

[13] Tobias Lang and Matthias Rettenmeier. 2017. Understanding consumer behavior with recurrent neural networks. In *Workshop on Machine Learning Methods for Recommender Systems*.

[14] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 59–66.

[15] Microsoft. 2019. LightGBM. https://lightgbm.readthedocs.io

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. (2013), 3111–3119.

[17] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[18] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[19] Humphrey Sheil, Omer Rana, and Ronan G. Reilly. 2018. Predicting Purchasing Intent: Automatic Feature Learning using Recurrent Neural Networks. *CoRR* abs/1807.08207 (2018).

[20] Chu Wang, Lei Tang, Shujun Bian, Da Zhang, Zuohua Zhang, and Yongning Wu. 2019. Reference Product Search. arXiv:arXiv:1904.05985

[21] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. *arXiv preprint arXiv:1902.04864* (2019).

[22] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2018. Session-based Recommendation with Graph Neural Networks. *arXiv preprint arXiv:1811.00855* (2018).

[23] Yuan Xia, Jingbo Zhou, Jingjia Cao, Yanyan Li, Fei Gao, Kun Liu, Haishan Wu, and Hui Xiong. 2019. Intent-Aware Audience Targeting for Ride-Hailing Service. In *Machine Learning and Knowledge Discovery in Databases*, Ulf Brefeld, Edward Curry, Elizabeth Daly, Brian MacNamee, Alice Marascu, Fabio Pinelli, Michele Berlingerio, and Neil Hurley (Eds.). Springer International Publishing, Cham, 136–151.

[24] Matei Zaharia, Reynold Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: a unified engine for big data processing. *Commun. ACM* 59 (2016), 56–65.

[25] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 207–212.

[26] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1079–1088.

A Framework for Recommender Systems Based on a Finite Multidimensional Model Space

Leonhard Seyfang
Research Unit of E-Commerce, TU Wien
Vienna, Austria
seyfang@ec.tuwien.ac.at

Julia Neidhardt
Research Unit of E-Commerce, TU Wien
Vienna, Austria
neidhardt@ec.tuwien.ac.at

ABSTRACT

In this conceptual paper we suggest a framework for flexible and efficient recommender systems. It is based on an unified finite multivariate model space for both user and products. Association functions map each entity to each model-dimension fuzzily. Finally distance- and learning-operations allow efficient operation. The main differences to existing approaches are the reduced model space and the fuzzy location of entities. The reduced model space is most advantageous where item features are inconsistent structured or sparse. The association function allows to express a *distribution* of agreement, not just a single location.

CCS CONCEPTS

• **Information systems → Personalization**; **Recommender systems**; *Collaborative search*; *Similarity measures.*

KEYWORDS

recommendation, personalization, feature based recommendation, similarity measurement, fuzzy mapping

1 INTRODUCTION

Tourism is for many reasons an interesting and challenging field for recommender systems: Travel experiences are complex and include various physical and mental aspects. Decisions are mainly based on subconscious, abstract ideas and emotions attached to them. At the same time hard constraints, like the available time frame and budget, have to be met. Also multiple persons are usually involved in the decision finding process. Products are very diverse, they are often inconsistent and incomplete documented. More often than not, products themselves do not satisfy the tourists need directly, but are prerequisites for the tourists dreams to be fulfilled. With all that challenges in mind, we reach for a flexible generic solution.

Generally, recommender systems aim to provide useful suggestions to their users. They use any combination of user-, item-, and context- information.

We suggest a recommendation-framework that:

- Reduces the feature-space to few interpretable (user-related) and manageable dimensions.
- Maps users and products, and other entities of interest to the model space.
- Treats the entity-dimension-relationship fuzzily.
- Provides a heuristic to efficiently compute distances between entities.
- Provides self-learning procedures in near real-time.

2 CORE CONCEPTS

In this section we introduce the essential concepts in theory. Practical aspects will be treated in section 3 and 4.

2.1 Model Space

In this framework we use a multidimensional, finite model space. All entities, *users*, *products*, or whatever abstract or actual items are of interest, are fuzzy–located in the very same model space. In most cases the number and interpretation of the dimensions will be defined domain specific. This can be done through domain-knowledge or by dimension reduction techniques such as factor analysis (see [3] for a related approach). The latter of course requires a suitable data corpus. For tourism seven factors have already been identified [5], [4].

Alternatively a generic, user oriented data model can be used to obtain a cross-domain recommender system. For example the Big Five personality traits [2] could be used straightforward as dimensions. For a comprehensive work on cross-domain recommendations see [1], and for thoughts on personality and recommender systems see [6].

2.2 Association Function

Association functions express the degree of accordance between entities and model-dimensions. They are most comparable to membership functions in fuzzy logic but should not be confused with probability density functions. Dimensions are treated independently, so each entity has a separate association function for each dimension.

In our model space, we think of each dimension as closed interval between 0 and 1. We believe that placing an entity on a single point on each dimension is an oversimplification. Instead it should be possible to express the spread of conformity over an adjustable range. Hence we were looking for a function that:

- Is defined on the closed interval $[0, 1]$;
- Takes values between 0 and 1;
- Is continuous (sufficiently small changes in x result in arbitrarily small changes in $f(x)$);
- Allows to specify location and dispersion independent of each other, hence takes (at least) two parameters;
- Is memory-efficient (is specified by as little as possible parameters).

We found the association function defined in (equation 1) fulfilling all requirements above.

$$f_{a,b}(x) = \begin{cases} 1 & \text{if} \quad a = b = 0 \\ \dfrac{x^a(1-x)^b}{\left(\frac{a}{a+b}\right)^a \left(1-\frac{a}{a+b}\right)^b} & \text{otherwise} \end{cases} \tag{1}$$

f is fully specified by two real parameters $a \geq 0$ and $b \geq 0$. An

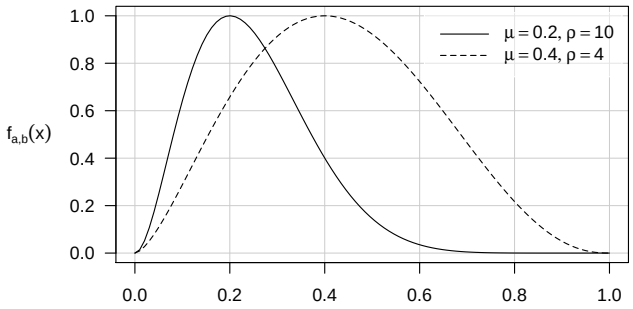

Figure 1: Two examples of association functions. Solid line: $\mu = 0.2$, $\rho = 10$, $a = 2$, $b = 8$; Dashed line: $\mu = 0.4$, $\rho = 4$, $a = 1.6$, $b = 2.4$;

alternative, more human comprehensible parametrization is given by the *location* parameter $\mu \in [0, 1]$ and the *precision* parameter $\rho \geq 0$. Both parametrizations can easily be converted into each other via (2), (3), (4), and (5). Examples for f are shown in figure 1.

$$\mu = \frac{a}{a+b} \qquad a + b > 0 \tag{2}$$

$$\rho = a + b \tag{3}$$

$$a = \mu\rho \tag{4}$$

$$b = (1-\mu)\rho \tag{5}$$

The value of $f_{a,b}(x)$ is in $[0, 1]$ for all valid a, b, and $x \in [0, 1]$. If $a = 0$ and $b = 0$, $f(x)$ is constant 1. We call $f_{0,0}$ the non-informative case. μ is not defined in the non-informative case and not needed either. Note: $f_{a,b}$ is proportional to the beta distribution $Beta(a + 1, b + 1)$, but density functions are scaled to an *area* of 1 while the association function is scaled to the *range* of $[0, 1]$. Further, Beta(0.5, 0.5) is called the *non-informative prior* in the context of Bernoulli trials in Bayesian statistics. Our case $f_{0,0}$ is not intended to possess the same *non-informativeness* and should not be confused.

Realistically ρ should not be to small since f gets increasingly vague as ρ approaches 0. On the other hand, ρ should not be to large neither as it would suggest an non-existing precision.

There are several ways how an entity gets its association functions:

(1) **Per mapping-algorithm**: For products, or whatever entities are considered for recommendations, mapping functions can be defined. A mapping function translates the available feature description into association function. Mapping algorithms can also be used related to users: in [5] users are mapped according to pictures they have selected. Also a mapping based on demographic features is possible.

(2) **Manually**: The graph of f can be used to set up an easy to use human interface. While using two sliders, one for the mode and one for the precision, one could alter the association function until the desired properties are reached. This option is favorable if no mapping-algorithm exists. In cases where the recommendation is in the foreground, it might be attractive to offer a tool for user-self-classification.

(3) **Self-learning**: Entities – typically users – can *learn* their position in the model space based on interaction with other entities – typically products – that already have been classified (see 2.4 for details).

The association function can also be used to *retrieve* item properties, particularly after a self-learning phase.

2.3 Distance

We define the distance $\boldsymbol{d}$ between two association functions as

$$\boldsymbol{d}(f_{a_1,b_1}, f_{a_2,b_2}) = \begin{cases} 0 & \text{if } \rho_1 = 0 \text{ or } \rho_2 = 0 \\ 1 - f_{a_1,b_1}(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{6}$$

where $\boldsymbol{x}$ is uniquely defined by the two properties (without loss of generality we assume from now on that $\mu_1 \leq \mu_2$):

$$\mu_1 \leq \boldsymbol{x} \leq \mu_2 \tag{7}$$

$$f_{a_1,b_1}(\boldsymbol{x}) = f_{a_2,b_2}(\boldsymbol{x}) \tag{8}$$

In words: $\boldsymbol{x}$ is the place between both modes where the two associ-

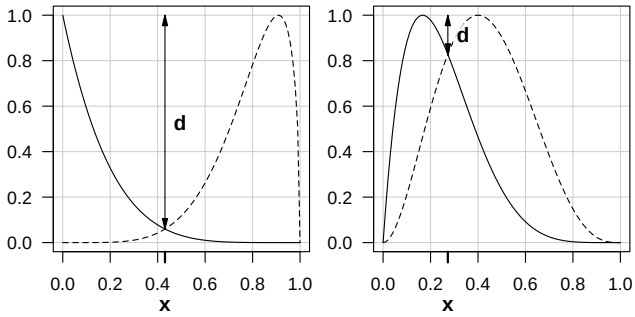

Figure 2: In the left example the association functions are very dissimilar hence the distance d is large. In the right example the association functions are somewhat similar so the distance is rather small.

ation functions intersect. $\boldsymbol{d}$ is 1 minus the value of f at $\boldsymbol{x}$. The basic idea of $\boldsymbol{d}$ is illustrated in figure 2.

Determining $\boldsymbol{x}$ requires numerical optimization but a good approximation is given by d:

$$d(f_{a_1,b_1}, f_{a_2,b_2}) = \begin{cases} 0 & \text{if } \rho_1 = 0 \text{ or } \rho_2 = 0 \\ 1 - \dfrac{f_{a_1,b_1}(\hat{x}) + f_{a_2,b_2}(\hat{x})}{2} & \text{otherwise} \end{cases} \tag{9}$$

with

$$\hat{x} = \frac{\mu_1 w_1 + \mu_2 w_2}{w_1 + w_2} \tag{10}$$

and

$$w_1 = \big(1 + 0.4\,(1 + s_1)\big)\sqrt{\rho_1} \tag{11}$$

$$w_2 = \big(1 + 0.4\,(1 - s_2)\big)\sqrt{\rho_2} \tag{12}$$

where s (skewness) is defined as

$$s = \frac{2(b-a)\sqrt{a+b+3}}{(a+b+4)\sqrt{(a+1)(b+1)}} \tag{13}$$

The closed solution for d is easy to compute and the deviation $|d - \boldsymbol{d}|$ is limited for a given range of ρ, e.g. $|d - \boldsymbol{d}| \leq 0.039$ for the reasonable assumption $0.5 \leq \rho \leq 10$ (without proof). Obvious properties of $\boldsymbol{d}$ are (also without proof):

$$\mu_1 = \mu_2 \;\Rightarrow\; \boldsymbol{d}(f_{a_1,b_1},\, f_{a_2,b_2}) = 0 \tag{14}$$

$$\boldsymbol{d}(f_{\mu_1,\rho_1}\, f_{\mu_2,\rho_2}) \;<\; \boldsymbol{d}(f_{\mu_1,\rho_1}\, f_{\mu_2+\epsilon,\rho_2}) \qquad \epsilon > 0 \tag{15}$$

$$\boldsymbol{d}(f_{\mu_1,\rho_1}\, f_{\mu_2,\rho_2}) \;>\; \boldsymbol{d}(f_{\mu_1,\rho_1}\, f_{\mu_2,\rho_2+\epsilon}) \qquad \epsilon > 0, \mu_1 \neq \mu_2 \tag{16}$$

The overall distance D between two entities is the weighted mean of the distances of all k dimensions.

$$D = \sum_{i=1}^{k} d_i v_i \tag{17}$$

The weights v are chosen proportional to the importance of the corresponding dimension.

2.4 Learning Procedure

The learning procedure allows entities (usually users) to adopt their location in the model space according to their interaction with other entities (usually products). It is based on the merge-operation.

The merge-operation m translates an ordered set of association functions F into a single association function:

$$F \xrightarrow{m} f_{a_{\text{new}},b_{\text{new}}} \tag{18}$$

We assume that no element of F is the non-informative function (otherwise those elements are simply removed as they do not hold information anyway). The cardinality of F (the number of elements in F) is denoted by n. The new parameter a_{new} is defined as

$$a_{\text{new}} \;=\; \begin{cases} 0 & \text{if} \quad n = 0 \\ a_1 & \text{if} \quad n = 1 \\ g\big(h(F)\big) \sum_{i=1}^{n} (a_i \mathbf{w}_i) & \text{if} \quad n > 1 \end{cases} \tag{19}$$

and b_{new} is defined accordingly.
Here $\mathbf{w}$ is a vector of weights associated with the elements of F with $\sum_{i=1}^{n} \mathbf{w}_i = 1$. h is a function that represents the dissimilarity of F. We currently use the mean of all pairwise distances within F for h (see equation 20) but other definition are certainly possible.

$$h(F) = \frac{1}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbf{w}_i \mathbf{w}_j} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Big(d(f_i, f_j)\, \mathbf{w}_i \mathbf{w}_j \Big) \tag{20}$$

The function g transforms the result of h to a reasonable shrinking factor, such as

$$g = \big(1 - h(F)\big)^{\lambda} \tag{21}$$

where $\lambda \geq 0$ is a tuning parameter. For larger lambdas the penalty for the dissimilarity increases. If $\lambda = 0$ there is no shrinking at all. In this case a_{new} and b_{new} are simply the weighted averages of the input-parameters (figure 3, left side). With a sufficient shrinkage factor on the other hand, m acts more like an union operation (figure 3, right side). Note that *shrinking* refers to a and b and consequently to the precision ρ whereas the spread of f works in the opposite direction. The merge-operation is commutative but generally not associative:

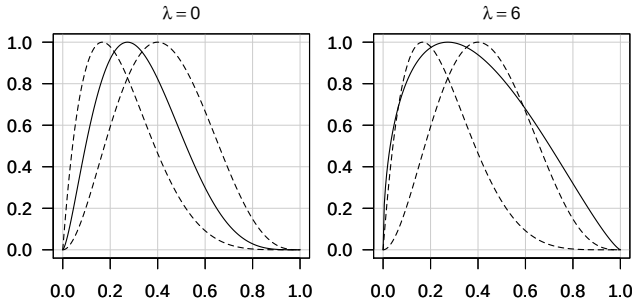

Figure 3: The tuning parameter λ controls the extent to which the dissimilarity h diminishes a_{new} and b_{new}. On the left $\lambda = 0$ and the precision parameter of the resulting function is simply the average of the precision parameters of the input functions (dashed line). On the right $\lambda = 6$ and the resulting function covers roughly the same area which is covert by the two input functions in conjunction.

$$m(f_{a_1,b_1},\, f_{a_2,b_2}) \;=\; m(f_{a_2,b_2},\, f_{a_1,b_1}) \tag{22}$$

$$m\big(m(f_{a_1,b_1},\, f_{a_2,b_2}),\, f_{a_3,b_3}\big) \;\neq\; m\big(f_{a_1,b_1},\, m(f_{a_2,b_2},\, f_{a_3,b_3})\big) \tag{23}$$

3 USAGE

A standard application works as follows: The model space (the number and interpretation of the dimensions) would be determined based on expert knowledge or dimensionality reduction methods or both. As mentioned earlier, seven factors have already been determined for the scope of tourism [5], [4].

Once the model space is specified, mappings from item-descriptions to the model dimensions must be implemented (see section 2.2).

In tourism, items are very diverse, including travel packages, hotels, flights, events, sights, natural phenomena, destination, cities, forms of sport and many others. Some of them are real *products* meaning bookable, other are not. The latter are still important for recommender systems as they serve as connection to actual products. Sometimes strong intangible aspects such as culture-dependent attributions or emotional concepts are involved. (The decision process might roughly be like: *honeymoon + love + Europe → city of love → Paris → hotel → room / suite*, not right away to the hotel room.)

Users obtain their profile in a self-learning way as they interact with items (or even other users). Depending on the particular domain and application, interactions can include book-, buy-, like-, rate-, comment-, view-, listen-to-, read-, search-, compare-, and other actions. Using the learning procedure from section 2.4, defined interactions modify the users profile towards the items interacted with. To define relevant interactions can be straightforward in some cases and sophisticated in others.

The initial association functions might be: the non-informative association function, the (dimensionwise) grand mean, the contextual a priori association function (for example based on known or estimated demographic characteristics).

The recommendation service itself calculates distances between users and products, sorts the results, and holds a list of most appropriate items ready. Computations can be done on demand or in advance. Filters might be implemented additionally to meet the users constraints.

Implementing stochastic components can increase serendipity and diversity but destruct predictability and reproducibility.

4 WORKED EXAMPLE

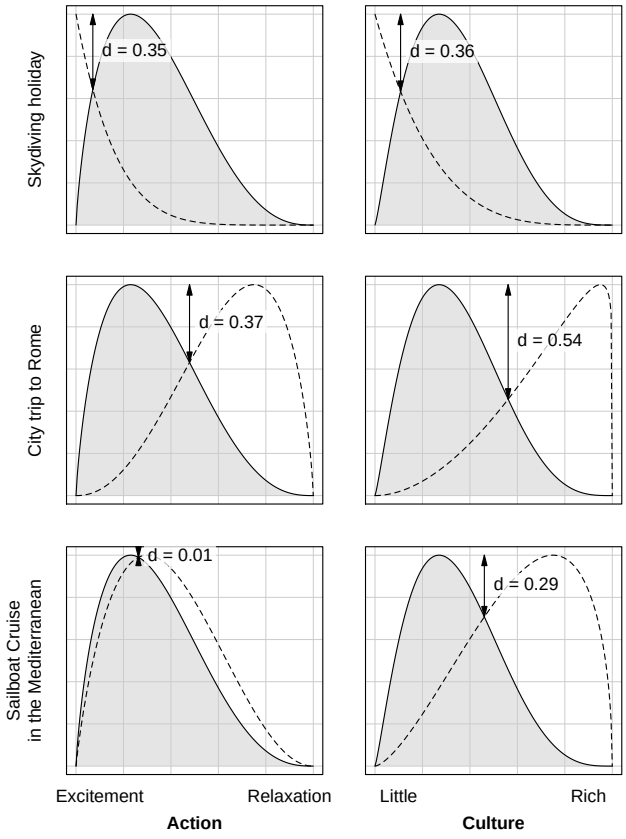

Figure 4: Example with two dimensions (columns) and three products (rows). The filled shapes display the users preferences, the dashed lines indicate the product properties.

For a simple example we assume that we have a travel recommendation system with two dimensions: Action and Culture, both equally important meaning equally weighted.

Our user is inclined towards exiting activities as long as they are not too extreme (figure 4, left column). The user is not really interested in culture (figure 4, right column).

We have three items to suggest: A skydiving holiday, a city trip to Rome, and a sailboat cruise in the Mediterranean.

The skydiving holiday is about as exiting as it gets with virtually no cultural options. (figure 4, first row).

The city trip to Rome offers ample cultural sights but besides that, it's not terribly exciting. (figure 4, second row).

Finally the sailboat cruise is exiting at times (although not as thrilling as skydiving), and the old Mediterranean cities also provide the opportunity to get in touch with old cultures. (figure 4, bottom row).

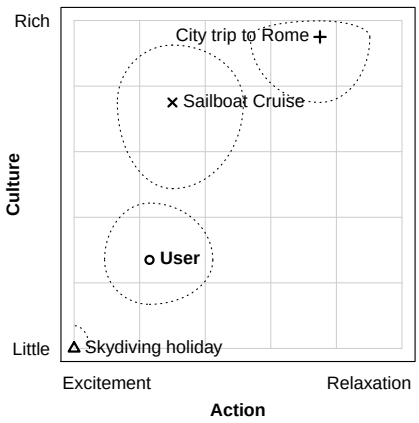

Figure 5: All item-locations in the R^2. The dotted lines are drawn approximately at $f = 0.75$ to indicate the spread along both dimensions.

In this toy example, the Mediterranean sailboat cruise would clearly be the best recommendation according to our measurement D (see equation 17), followed by the skydiving holiday. However if we had used the location parameter μ in conjunction with the Euclidean distance or the Manhattan distance, the skydiving holiday would have appeared to be the closest to the user. The reason for this divergence is the different spread of associations.

In table 1 all user-item distances are presented, according to Euclidean-, Manhattan-, and D-distance. Figure 5 illustrates the locations of all items in the R^2.

Table 1: Euclidean-, Manhattan-, and D-distance for all items.

Item	Euclidean	Manhattan	D
Skydiving	0.35	0.50	0.36
Rome	0.86	1.20	0.45
Sailboat Cruise	0.49	0.55	0.15

5 DISCUSSION

The framework presented here offers interesting possibilities as it is flexible, possibly cross-domain, self-learning, and the entity-dimension-memberships relation is easy to understand. It has no cold start problem with new items and it is not necessary to match an user to other similar users. It can serve as basis for multivariate outlier detection and for cluster analysis. Deviations in the product- and user- distribution can be revealed as side effect.

However this approach comes with two downsides: Firstly the dimensions of the model-space must be defined in advance and are hard to modify in a running system. Hence setting up the model

space is the crucial task. Secondly the mapping from the original feature space to the model dimensions must be implemented. Manual input is simple but time-consuming thus expensive with large quantities. The next steps will be the utilization in an operating recommender system and measuring and reporting the performance, ideally in comparison with an established system.

REFERENCES

[1] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. 2015. *Recommender Systems Handbook* (2 ed.). Springer, New York Heidelberg Dordrecht London, Chapter 27, 919–959.

[2] Oliver P. John and Sanjay Srivastava. 2008. *Handbook of Personality: Theory and Research* (3 ed.). The Guilford Press, New York, Chapter 4, 114–158.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42 (08 2009), 30–37.

[4] Julia Neidhardt, Rainer Schuster, Leonhard Seyfang, and Hannes Werthner. 2014. Eliciting the Users' Unknown Preferences. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 309–312.

[5] Julia Neidhardt, Leonhard Seyfang, Rainer Schuster, and Hannes Werthner. 2015. A picture-based approach to recommender systems. *Information Technology & Tourism* 15-1 (2015), 49 – 69.

[6] Marko Tkalcic and Li Chen. 2015. *Recommender Systems Handbook* (2 ed.). Springer, New York Heidelberg Dordrecht London, Chapter 21, 715–739.

TourWithMe: Recommending peers to visit attractions together

Sebastián Vallejos
ISISTAN Research Institute
CONICET-UNICEN
Tandil, Buenos Aires, Argentina
sebastian.vallejos@isistan.unicen.edu.ar

Marcelo G. Armentano
ISISTAN Research Institute
CONICET-UNICEN
Tandil, Buenos Aires, Argentina
marcelo.armentano@isistan.unicen.edu.ar

Luis Berdun
ISISTAN Research Institute
CONICET-UNICEN
Tandil, Buenos Aires, Argentina
luis.berdun@isistan.unicen.edu.ar

ABSTRACT

When a user travels alone or in a small group, usually likes to share the experience of visiting different attractions in a larger group. This article propose TourWithMe, our first approach to the problem of recommending peers to visit attractions in a city together. To this aim, TourWithMe automatically learns the user's interests from previously visited attractions, that are then combined with explicit preferences provided by the user to find compatible tourists in the city. TourWithMe recommends to the user different groups and, for each group, attractions that they would enjoy visiting together.

CCS CONCEPTS

• **Information systems → Recommender systems**; **Social recommendation**; *Crowdsourcing*.

KEYWORDS

group recommender system; tourism; crowdsourcing; user modeling

1 INTRODUCTION

Visiting a new city is always a challenging experience. Among the set of touristic attractions available in the city, tourists have to select, and usually prioritize, those that are more appealing according to their interests, available time and budget. In consequence, planning a holiday is usually a stressful activity and travellers relay in the use of different applications that may support their decision-making processes.

Recommender systems for tourism arisen to cope with the information overload to which tourists face when visiting a new city. In this regard, recommender systems have focused on different aspects of the domain, such as recommending hotels [1, 25], routes [10, 16], restaurants [9], itineraries [7, 15], and attractions [13, 33, 34].

A hot topic in recommender systems research is the recommendation of items to groups of users, since recommendations need to satisfy a group of users as a whole, instead of individual users [5, 6]. In the field of tourism, recommender systems for groups have been proposed for users who travel with a predefined group (for example, a group of friends or family travelling together) [2, 11].

To the best of our knowledge, none of the existing approaches considered the proposal of groups to visit different attractions together. This kind of recommender system might be extremely useful for users who visit a destination alone or in a small group (for example, with his/her couple) and who want to meet peers to share the experience of touring together. The need of this kind of service becomes clear given the existence of many websites [1,2,3] and social network groups [4,5,6] dedicated to people who wants to meet other people and form groups for tourism.

In this context, the popularization of mobile devices brings forward new challenges and opportunities for the implementation of personalized applications and location-aware services. Particularly, mobile devices enable to capture the user's mobility history and taking advantage of geographic proximity of other users to enhance the user experience [14].

In this article, we present TourWithMe, a recommender system in the tourism domain that takes advantage of mobile devices for recommending travellers to form groups to visit attractions or points of interest (POI) together. Our approach considers geolocalization provided by mobile devices in two ways. On the one hand, the approach implicitly learns the user's interest from the places he/she visits, the amount of time spent in each place, and the time spent travelling to those places. In this way, users do not have to manually check-in every place they visit or to explicitly provide their interests, as required by most of the current approaches. On the other hand, the approach finds other tourists in the proximity of the user and suggests forming a group with those users who have similar interests. Once a group is formed, the approach suggests to visit nearby venues that the whole group would enjoy visiting.

The remainder of this paper is organized as follows. Section 2 discusses related works about recommenders system for tourism. Section 3 presents the proposed approach for recommending travellers forming groups to visit attractions together. Finally, Section 4 presents conclusions and future works.

2 RELATED WORK

Recommenders System for tourism is a hot topic that has been addressed in several works in the last years. These works proposed approaches to recommend users to visit a nearby POI or even a tour itinerary. To carry out this task, proposed approaches used different information, such as the user's current location, information about nearby POIs, user preferences and interests, current day and time, temporal restrictions, etc. The kind of information used and the way in which this information is obtained vary depending on the approach.

In [31] and [19], authors asked users to manually provide their interest and preferences. Both approaches recommend a personalized tour itinerary that fits the user's interests. To carry out this

[1] https://www.yourtravelmates.com/
[2] https://www.workaway.info/
[3] https://www.couchsurfing.com/
[4] https://www.facebook.com/groups/altmtl/
[5] https://www.facebook.com/groups/1157818554266712/
[6] https://www.facebook.com/groups/travellinks/

task, [31] used a Greedy algorithm while [19] used an evolutionary algorithm. The main disadvantages of these works are that manually introducing interests may be a stressful task for users, and they tend to be reluctant to explicitly provide this kind of information [26]. For this reason, some works in the literature proposed to automatically infer the user's interests by analyzing the previously visited POIs.

To address this task, some works used check-ins made by user in location based social networks (LBSN) [17, 35] and geotagged photos from social networks [4, 20, 21] in order to reconstruct the history of visited POIs. In [4, 17, 20], authors proposed approaches that infer the interests of the user for each POI category according to the number of visited POIs belonging to that category. These approaches use these interests to generate a ranking of possible POIs to be visited by the user. In [35], authors proposed a similar approach that infers the user's interest from Jiepang check-ins data. As the user's interests may change according to the time of day, this approach also divides the day into six time slots and calculates the user's interests for each time slot separately. In [21], authors proposed an approach that calculates the duration of each visit by considering the timestamps of the first and the last photos took in the visited POI. The approach uses this information to estimate the user interest for a POI category. For example, if the user spends more time in museums than the average time spent by other users, the approach infers that the user is interested in museums.

As some tourists tend to travel in group, recommending POIs to a group of users instead of to a single user is a useful feature in the tourism domain. Some approaches in the literature address this feature by combining the users' profiles into a single group profile [12, 27]. In this way, approaches designed for recommending POIs to a single profile (usually a user profile) can recommend also POIs to a group by taking the group profile as input. There are two main approaches to combine user profiles: aggregation, when the resultant group profile is the union of all the group members preferences; and intersection, when the resultant group profile is the intersection of all the group members preferences. The approach presented by [5] used an hybrid approach for generating recommendations to groups of tourists, which combines the demographic information of users, the ratings of the community and the content-specific information about the items. The individual ratings inferred from the hybrid profile are weighted according to a fixed set of social relationships among the members of the group. Finally, the influenced individual ratings of all members of a group are combined to estimate a group rating for different items.

To the best of our knowledge, none of the existing approaches considered the proposal of groups to visit different attractions together. The most similar approach to the one presented in this article is the one presented in [22]. In this work, authors proposed an approach oriented to assisting travel agencies for grouping tourists. The approach uses K-means algorithm to cluster a predefined set of users into K groups. Each resultant group contains users with similar interest. Then, the approach assigns a tour itinerary from a set of predefined tour itineraries to each group of users. However, this approach is not useful for a tourist who is alone in an unknown city and wants to meet peers to visit POIs together.

3 SYSTEM DESIGN

Figure 1 shows a high-level diagram of TourWithMe. As shown in the diagram, the approach consists of three steps. In the first step (A) the approach infers the user's interests from the geolocation data of the user. By knowing the POIs visited by the user, the time spent in each place, and the time spent travelling to those places it is possible to estimate the interest of the user in such places. This step is detailed in Section 3.1. In the second step (B), when a user requires it, the approach proposes forming a group with nearby users. The approach uses the profile information of each user to form a cohesive group of users with similar interests. In this sense, there is more chance of finding a POI that is attractive to everyone in the group. This step is detailed in Section 3.2. Finally, in the third step (C), the approach recommends the top-five POIs to the group by considering the interest information of each user in the group. This step is detailed in Section 3.3.

3.1 Inferring the user's interests

This step consists of analyzing the mobility data of the user in order to infer his/her preferences. In order to carry out this task, TourWithMe takes advantage of modern mobile devices. These devices are equipped with several sensors that allow estimating the location of the user. For example, it is possible to estimate the user location by knowing the nearby WiFis or by using the GPS of the smartphone. By tracking the user location, TourWithMe detects visits to places, also named stay points. A stay point is defined in the literature as a geographic region where the user stayed over a time threshold T_s within a distance threshold D_s [24, 29, 32]. In particular, TourWithMe detects a visit when the user stays for more than 5 minutes within a distance of 50 meters. Each visit is represented as a tuple (C, T_i, T_e), where C is the centroid of the geographic area where the user stayed, T_i is the start time of the visit and T_f is the end time of the visit.

When a visit is detected, TourWithMe identifies the POI visited by the user, if any. To carry out this task, TourWithMe relies on public data extracted from OpenStreetMap[7] (OSM). In particular, TourWithMe uses the Overpass Turbo API[8] to query POIs that are less than 50 meters away from the visit. If there are no nearby POI, it is considered that the user stayed in some other place (e.g. in a store). If there is more than one nearby POI, TourWithMe selects the POI with the highest score according to Equation 1. This equation compares the duration of a visit V of user U and the average time of visit for a POI P. The average time of visit for P is computed from previous visits of other users to the same POI. It is important to notice that the user can manually modify the visited POI if needed.

$$score(V, P) = 1 - \frac{|avgDurationOfVisit(P) - duration(V)|}{avgDurationOfVisit(P)} \quad (1)$$

Once the visit has an associated POI, TourWithMe estimates the interest of the user in that POI. The interest of the user in a POI is a real value between 0 and 1 where 0 means that the user is not interested in the POI and 1 corresponds to the maximum interest. This value is computed according to Equation 2 and considers the

[7] https://www.openstreetmap.org/
[8] http://overpass-turbo.eu/

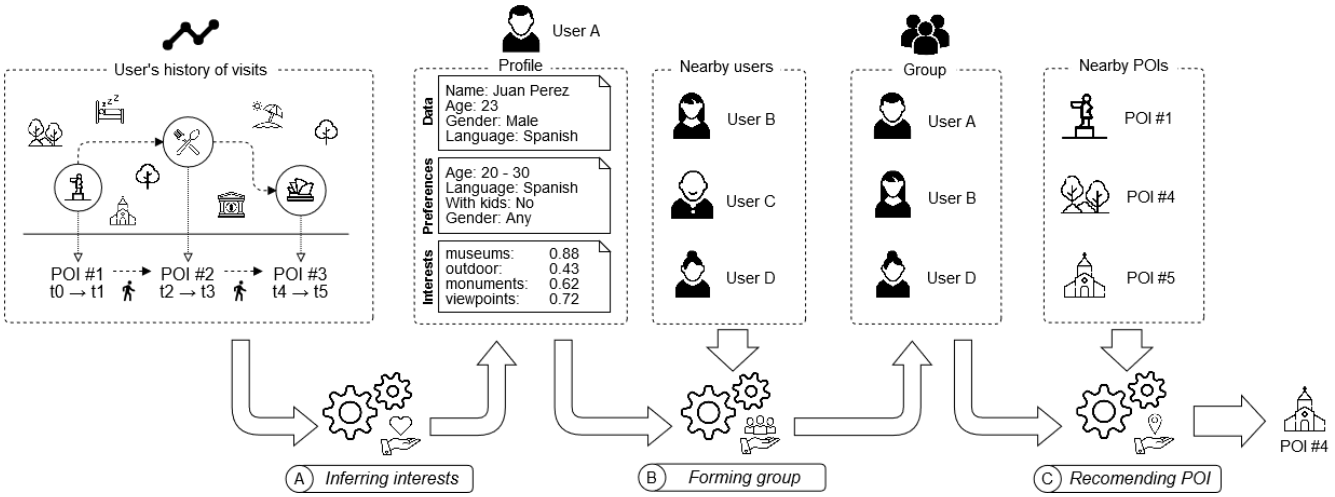

Figure 1: TourWithMe approach

time that the user spent in the POI ($int_{visit-time}$) and the time of the travel T to that POI ($int_{travel-time}$).

$$int(T, V, P) = \frac{int_{visit-time}(V, P) + int_{travel-time}(T, P)}{2} \quad (2)$$

To compute the first term of the equation, in [21] authors proposed to compute the ratio between the time spent by the user in the POI and the average duration of visits to that POI. However, this approach is not useful when a POI has different groups of users who visit the POI with different average times. For example, a museum can offer 1-hour and 2-hours guided tours. An average of 1.5 hours is then not representative for a user taking the 1-hour tour nor to a user taking the 2-hours tour. Furthermore, computing the interest of a user in a POI in this way doesn't give a normalized value of the user interest. To overcome the above-mentioned problems, TourWithMe uses the cumulative percentage of duration of visits. Equation 3 shows how the approach computes $int_{visit-time}(V, P)$ for a visit V to a POI P. For example, if spent 14 minutes in P, and 60% of people stayed less than 14 minutes in P, then the interest of the user in P is 0.6.

$$int_{visit-time}(V, P) = \frac{\sum_{d=0}^{duration(V)} V_{d,p}}{|V_p|} \quad (3)$$

where $V_{d,p}$ is number of visits to POI p with a duration d, and V_p is the number of visits to POI p.

The second term of Equation 2, $int_{travel-time}(T, P)$, compares the time spent by a user in a POI with respect to the time spent travelling to that POI. In [8] authors proposed travel-time ratio as a way to calculate how much time a user is willing to travel to perform an activity. In [30], authors found higher travel-time ratios for activities in which users are interested, such as sport and recreation activities. Mapping the conclusions arrived in the above-mentioned research to the tourism domain, we can assume that if a user travels a long time to visit a given POI, he/she has a great interest in that POI. Equation 4 details how to calculate this ratio for simple journeys in which the user goes to a POI and then returns to his place of lodging. The way to calculate the time ratio for journeys in which the user visit several POIs before returning his/her place of lodging is detailed in [30].

$$int_{travel-time}(T, V) = \frac{duration(T)}{duration(T) + duration(V)} \quad (4)$$

By knowing the interest of the user in each POI he/she visited, it is possible to estimate his/her interest for each POI category. As POIs are extracted from OSM, they have different pairs of key-value describing them. For example, {"$tourism$" : "$museum$"}, {"$name$" : "$Le\ Louvre$"}. These pairs of key-value are used to label the POI with POI categories. For example, "Le Louvre" is categorized as a "museum". To calculate the interest of a user for a specified POI category C, TourWithMe calculates the average interest of the user in every POI p belonging to C that he/she previously visited (Equation 5).

$$int_{inferred}(U, C) = \frac{\sum_{p \in C} interest(U, p)}{|C|} \quad (5)$$

3.2 Forming groups

For suggesting groups to a user, TourWithMe considers three factors: geolocalization, user's preferences and similarity between users' interests regarding POIs categories. When the user asks for suggestions or when he/she arrives to a new city, TourWithMe first find the set of users S_R within a parameter radio R from the user's current location. If R is not set by the user, TourWithMe considers the set of users visiting the same city. S_R contains then the set of candidate users near to the user's location.

Once the set of candidate users is obtained, it is filtered by the user's preferences. User's preferences are a list of restrictions that the user is able to manually fill in his/her profile, and indicate the system what kind of users are expected to be recommended to the target user. These preferences, which are all optional, include:

- age range: indicates the minimum and maximum age of other users in the group
- sex: preferred sex of people in the group (male, female, any)

- languages: a list of languages that users in the group should speak
- country of residence: if the user prefers other users from specific countries
- children preference: users can indicate whether they prefer tourists traveling with children or not.

Then, if a user established in his/her profile that he/she prefers other tourists aging between 20 and 30, any candidate whose age is outside those limits is removed from the set of candidates. The resulting set S_f contains the set of compatible candidates with the user's preferences.

Other kind of preferences included in the user profile are the following:

- a list of categories of interest: an explicit list of the POI categories in which the user manually indicated interest. Categories are taken from the OpenStreetMap Semantic Network [3].
- budget: indicates the amount of money the user expects to spend while visiting attractions. This variable is discretized in four values (0, $, $$, $$$), indicating free, cheap, moderate, and expensive POIs, respectively

The list of categories manually defined by the user and the inferred interests (which were obtained as described in Section 3.1) are combined to define the real interest of a user U in a category C (Equation 6). If user U explicitly indicated interest in C (by adding it to his/her list of interests), then $int(U, C)$ is the average between 1 and $int_{inferred}(U, C)$. Otherwise, $int(U, C)$ is equals to $int_{inferred}(U, C)$.

$$int(U,C) = \begin{cases} \frac{1+int_{inferred}(U,C)}{2}, & \text{if } U \text{ is interest in C} \\ int_{inferred}(U,C), & \text{otherwise} \end{cases} \quad (6)$$

In the current implementation of TourWithMe, each candidate user v in S_f is ranked by computing the soft cosine similarity with respect to the target user U (Equation 7). This similarity measure does not assume that features in the space model are independent and then introduce the similarity of features into the equation of the traditional cosine similarity.

$$soft\_cosine(U,v) = \frac{\sum_{i,j}^{N} s_{ij} U_i v_j}{\sqrt{\sum_{i,j}^{N} s_{ij} U_i U_j} \sqrt{\sum_{i,j}^{N} s_{ij} c_i v_j}} \quad (7)$$

where U_i is the ith feature for user U, v_i is the ith feature for user v, and s_{ij} is the similarity between the ith and the jth features. The similarity between features i and j, s_{ij}, is computed by using the semantic similarity of OSM tags [3]. The set $S_C \subset S_f$ with the K most similar users is considered for forming groups in the next step.

When a user U asks for a group recommendation, he/she must define a preferred group size Z (where $Z < K$). Then, from S_c, all possible groups of size Z including the target user U are computed, and a cohesion score is assigned to each of them. Cohesion is computed as the average similarity between each pair of users in the group. Groups are finally sorted by the cohesion score.

3.3 Recommending POIs

Although groups are formed by finding tourists with similar interests, different users always will have some different interests. To address these diverse interests, most approaches in the literature build a group interest profile by aggregating or by intersecting the preferences of all group members [11, 13, 27, 28]. From these two options, aggregating preferences is preferable since it allows introducing serendipity in the recommendations enabling the user to discover attractions that may not be recommended by a recommender system for individuals. Serendipitous items are items that users would not find by themselves or even look for, but that would enjoy consuming. The introduction of serendipity in recommender systems is fundamental to avoid users losing the interest in recommendations due to a overspecialization of the system in the user's already-known interests [18]. This overspecialization, avoids the recommender system to learn new interests of the user, and enables the user to be able to predict by themselves what items would be recommended by the system, reducing in consequence the user's satisfaction with the recommendations.

For example, Figure 2 shows a group of three users with their respective interests. By aggregating user interests, the interest of the resultant group profile in a category C_i is the average interest of the three users in C_i. In the literature, this is known as *average aggregating strategy* [23]. As the interest of user B in C_2 is not defined, the interest of the whole group in C_2 is calculated by considering only users A and C. Thus, the resultant group profile has a high interest in category C_2. In this way, if the approach recommends a POI of C_2, it will encourage User B to visit a new kind of POI. Instead, by intersecting user interests, the resultant group profile will not have any interest value defined for C_2, since not all users of the group have an interest defined in C_2. Thus, the approach will encourage users to continue visiting the same kind of POIs they already visited before.

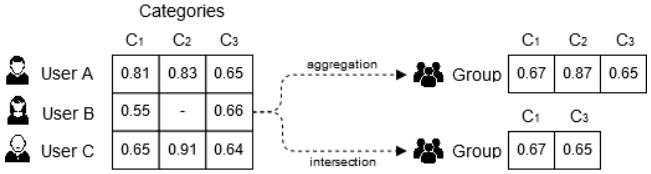

Figure 2: Aggregation vs. intersection of interests

TourWithMe builds a group interest profile based on the average interest preference of all group members. Given a group $g = u_1, ..., u_k$, the group interest in a cagetory c is defined according to Equation 8.

$$int(g,c) = \frac{1}{|g_c|} \sum_{u \in g_c} int(u,c) \quad (8)$$

where $g_c \subset g$ are the members of g for which the interest $int(u, c)$ is defined.

Then, the interest of a group g in a given POI p is computed according to Equation 9.

$$int(g,p) = \frac{\sum_{c \in C_p} int(g,c)}{|C_p|} \quad (9)$$

where C_p are the categories associated to POI p.

Continuing with the example of Figure 2, by using the average interest of all the group members not necessary may lead to making the best recommendation. For example, Figure 2 shows that the group profile has an interest of 0.67 for C_1 and 0.65 for C_3. Thus, recommending a POI belonging to C_1 would be preferred than recommending a POI belonging to C_3. However, the variation of interests for this category is very high: User A has an interest of 0.81, while User B has an interest of 0.55. Thus, visiting a POI belonging to C_1 seems to be *unfair* for User B. Moreover, User A will want to stay in the POI a longer time, while user B will want to leave before. Instead, when visiting a POI belonging to C_3, the three users will have a similar interest in the POI and there are more chances that they will agree about how long to be in that place.

To considering this situation, TourWithMe looks for recommending the POI that best fits the group profile at the same time that it reduces the variation of interest among users for the recommended POI. Equation 10 shows how TourWithMe score a POI p for a group g. All POIs in the user's neighbourhood are ranked according to this equation, and the top-5 POIs are assigned to each group as recommendations.

$$score(g,p) = int(g,p) - \frac{maxInterest(g,p) - minInterest(g,p)}{|g|} \quad (10)$$

where $maxInterest(g,p) - minInterest(g,p)$ is the maximum variation of interest between the members of group g for POI p.

Along to each recommendation TourWithMe computes the estimated time that the group would spend in each POI by using the cumulative percentage of duration of visits, as detailed in Equation 3. In this case, if the group interest in the category of a POI p is, for example, 0.6 and the time spent by 60% of the people at the given POI is t, we assign t as the estimated time that the group would spend at p.

4 CONCLUSIONS AND FUTURE WORK

In this article we presented TourWithMe, a first approach to the problem of recommending peers to visit different attractions in a group. We believe that our approach might appeal tourists traveling alone or in small groups to enhance the experience of enjoying the attractions offered by a new city.

TourWithMe is currently in a prototype stage, and is developed as a native Android application. This application tracks user location and detect visits when the user stays for more than 5 minutes within a distance of 50 meters. Then, TourWithMe associates each visit to a POI extracted from OpenStreetMap when possible. In addition, TourWithMe identifies the transport mode of each travel, which in the future may be a useful feature for POI recommendation. For example, if user moves by car, it is possible to recommend more distant POIs than if he/she moves on foot.

The next step in our research is to evaluate our approach with a benchmark dataset. As there is no benchmark dataset available for POI recommendation for group of users, most works in the literature use datasets with individual ratings and simulate groups. The rating of a simulated group for a POI may be estimated as the average ratings of the group members. The main challenge after evaluating the proposed approach with a simulated dataset will naturally be the validation with real users.

REFERENCES

[1] Marie Al-Ghossein, Talel Abdessalem, and Anthony Barré. 2018. Cross-Domain Recommendation in the Hotel Sector. In *Proceedings of RecTour 2018*, Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, and Markus Zanker (Eds.). 1–6.

[2] Aris Anagnostopoulos, Reem Atassi, Luca Becchetti, Adriano Fazzone, and Fabrizio Silvestri. 2016. Tour recommendation for groups. *Data Mining and Knowledge Discovery* 31, 5 (sep 2016), 1157–1188. https://doi.org/10.1007/s10618-016-0477-7

[3] Andrea Ballatore, Michela Bertolotto, and David C. Wilson. 2012. Geographic knowledge extraction and semantic similarity in OpenStreetMap. *Knowledge and Information Systems* 37, 1 (oct 2012), 61–81. https://doi.org/10.1007/s10115-012-0571-0

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

[5] Ingrid Christensen, Silvia Schiaffino, and Marcelo Armentano. 2016. Social group recommendation in the tourism domain. *Journal of Intelligent Information Systems* 47, 2 (mar 2016), 209–231. https://doi.org/10.1007/s10844-016-0400-0

[6] Amra Delic, Julia Neidhardt, Thuy Ngoc Nguyen, and Francesco Ricci. 2018. An observational user study for group recommender systems in the tourism domain. *Information Technology & Tourism* 19, 1-4 (feb 2018), 87–116. https://doi.org/10.1007/s40558-018-0106-y

[7] Linus W Dietz and Achim Weimert. 2018. Recommending Crowdsourced Trips on wOndary. In *Proceedings of RecTour 2018*, Julia Neidhardt, Wolfgang Wörndl, Tsvi Kuflik, and Markus Zanker (Eds.). 13–17.

[8] Martin Dijst and Velibor Vidakovic. 2000. Travel time ratio: the key factor of spatial reach. *Transportation* 27, 2 (2000), 179–199. https://doi.org/10.1023/a:1005293330869

[9] Haoxian Feng and Thomas Tran. 2018. Context-Aware Approach for Restaurant Recommender Systems. In *Encyclopedia of Information Science and Technology, Fourth Edition*. IGI Global, 1757–1771. https://doi.org/10.4018/978-1-5225-2255-3.ch153

[10] Ander Garcia, Maria Teresa Linaza, Olatz Arbelaitz, and Pieter Vansteenwegen. 2009. Intelligent Routing System for a Personalised Electronic Tourist Guide. In *Information and Communication Technologies in Tourism 2009*. Springer Vienna, 185–197. https://doi.org/10.1007/978-3-211-93971-0_16

[11] Inma Garcia, Laura Sebastia, and Eva Onaindia. 2011. On the design of individual and group recommender systems for tourism. *Expert Systems with Applications* 38, 6 (jun 2011), 7683–7692. https://doi.org/10.1016/j.eswa.2010.12.143

[12] Inma Garcia, Laura Sebastia, Eva Onaindia, and Cesar Guzman. 2009. A Group Recommender System for Tourist Activities. In *E-Commerce and Web Technologies*. Springer Berlin Heidelberg, 26–37. https://doi.org/10.1007/978-3-642-03964-5_4

[13] Damianos Gavalas and Michael Kenteris. 2011. A web-based pervasive recommendation system for mobile tourist guides. *Personal and Ubiquitous Computing* 15, 7 (may 2011), 759–770. https://doi.org/10.1007/s00779-011-0389-x

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

[15] 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 (mar 2014), 291–328. https://doi.org/10.1007/s10732-014-9242-5

[16] Koji Kawamata and Kenta Oku. 2019. Roadscape-based Route Recommender System Using Coarse-to-fine Route Search. *Journal of Information Processing* 27, 0 (2019), 392–403. https://doi.org/10.2197/ipsjjip.27.392

[17] K. Kesorn, W. Juraphanthong, and A. Salaiwarakul. 2017. Personalized Attraction Recommendation System for Tourists Through Check-In Data. *IEEE Access* 5 (2017), 26703–26721. https://doi.org/10.1109/access.2017.2778293

[18] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2016. Challenges of Serendipity in Recommender Systems. In *Proceedings of the 12th International Conference on Web Information Systems and Technologies*. SCITEPRESS - Science and and Technology Publications. https://doi.org/10.5220/0005879802510256

[19] Yohei Kurata and Tatsunori Hara. 2013. CT-Planner4: Toward a More User-Friendly Interactive Day-Tour Planner. In *Information and Communication Technologies in Tourism 2014*. Springer International Publishing, 73–86. https://doi.org/10.1007/978-3-319-03973-2_6

[20] 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 Press. https://doi.org/10.1145/2744680.2744693

[21] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2017. Personalized trip recommendation for tourists based on user interests,

points of interest visit durations and visit recency. *Knowledge and Information Systems* 54, 2 (may 2017), 375–406. https://doi.org/10.1007/s10115-017-1056-y

[22] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and S hanika Karunasekera. 2016. Towards next generation touring: Personalized group tours. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*. 421–430.

[23] Judith Masthoff. 2015. Group Recommender Systems: Aggregation, Satisfaction and Group Attributes. In *Recommender Systems Handbook*. Springer US, 743–776. https://doi.org/10.1007/978-1-4899-7637-6_22

[24] Raul Montoliu and Daniel Gatica-Perez. 2010. Discovering human places of interest from multimodal mobile phone data. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia - MUM '10*. ACM Press. https://doi.org/10.1145/1899475.1899487

[25] Qing Qi, Jian Cao, Yudong Tan, and Quanwu Xiao. 2018. Cross-Domain Recommendation Method in Tourism. In *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE. https://doi.org/10.1109/pic.2018.8706265

[26] Feng Qiu and Junghoo Cho. 2006. Automatic identification of user interest for personalized search. In *Proceedings of the 15th international conference on World Wide Web - WWW '06*. ACM Press. https://doi.org/10.1145/1135777.1135883

[27] Logesh Ravi and Subramaniyaswamy Vairavasundaram. 2016. A Collaborative Location Based Travel Recommendation System through Enhanced Rating Prediction for the Group of Users. *Computational Intelligence and Neuroscience* 2016 (2016), 1–28. https://doi.org/10.1155/2016/1291358

[28] Senjuti Basu Roy, Laks V.S. Lakshmanan, and Rui Liu. 2015. From Group Recommendations to Group Formation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15*. ACM Press. https://doi.org/10.1145/2723372.2749448

[29] Xiao Wen Ruan, Shou Chung Lee, and Wen Chih Peng. 2014. Exploring Location-Related Data on Smart Phones for Activity Inference. In *2014 IEEE 15th International Conference on Mobile Data Management*. IEEE. https://doi.org/10.1109/mdm.2014.71

[30] Yusak O. Susilo and Martin Dijst. 2010. Behavioural decisions of travel-time ratios for work, maintenance and leisure activities in the Netherlands. *Transportation Planning and Technology* 33, 1 (feb 2010), 19–34. https://doi.org/10.1080/03081060903429280

[31] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. 2011. The City Trip Planner: An expert system for tourists. *Expert Systems with Applications* 38, 6 (jun 2011), 6540–6546. https://doi.org/10.1016/j.eswa.2010.11.085

[32] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. 2009. Mining Individual Life Pattern Based on Location History. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE. https://doi.org/10.1109/mdm.2009.11

[33] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. 2017. Spatial-Aware Hierarchical Collaborative Deep Learning for POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 29, 11 (nov 2017), 2537–2551. https://doi.org/10.1109/tkde.2017.2741484

[34] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. 2016. Adapting to User Interest Drift for POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (oct 2016), 2566–2581. https://doi.org/10.1109/tkde.2016.2580511

[35] Zhiwen Yu, Huang Xu, Zhe Yang, and Bin Guo. 2016. Personalized Travel Package With Multi-Point-of-Interest Recommendation Based on Crowdsourced User Footprints. *IEEE Transactions on Human-Machine Systems* 46, 1 (feb 2016), 151–158. https://doi.org/10.1109/thms.2015.2446953

Balancing Preferences, Popularity and Location in Context-Aware Restaurant Deal Recommendation: A Bristol, Cardiff and Brighton Case Study

Ercan Ezin, Hugo Alcaraz-Herrera
{ercan.ezin,h.alcarazherrera}@bristol.ac.uk
University of Bristol
Bristol, United Kingdom

Iván Palomares
i.palomares@bristol.ac.uk
University of Bristol. Bristol, United Kingdom
The Alan Turing Institute. London, United Kingdom

ABSTRACT

We propose a personalisation solution to recommend tailored restaurant deals for residents or visitors in a city. Unlike previous work on recommendations in the restaurant sector where actual venues are recommended, we focus on suggesting specific products in the form of deals offered by such restaurants. This is done by jointly filtering relevant information for the end-user based on their food-drink preferences, the popularity of the restaurant, its proximity to the user's location and temporal constraints on the availability of deals. A real case study has been conducted upon datasets provided by Wriggle, a platform for discovering local deals in various cities across England.

KEYWORDS

Personalised Tourism, Restaurant Recommendation, Preference Modeling, Context-Aware Recommendation, Weighting

1 INTRODUCTION

Personalisation services for tourism, leisure and entertainment have been investigated for recommending Points-of-Interest (PoIs) or sequences of them [1, 2], selecting suitable cities for a group itinerary [3], or recommendations in the hotel sector [4, 6], to name a few. This study focuses on recommendations in the restaurant sector, which has also attained significant attention within the tourism landscape: in medium to large cities where both residents and visitors alike search for new restaurants, cafes or bars amid hundreds or thousands of available options [7], eating or drinking out is a cornerstone activity where personalisation turns indispensable to help them finding venues that meet their taste.

Various research efforts have been made on recommending suitable restaurants based on different forms of user preferences and contextual factors [8–10]. However, these works typically focus on recommending venues, by analysing characteristics associated to the restaurant itself, without looking at specific products (e.g. dishes, drinks, deals, etc.) offered by that restaurant or analysing how they meet the specific user needs or preferences. Despite this is an important decision-making step for for customers, many of them also seek specific dishes or suitable offers/deals that meet their preferences to a deeper level of granularity. To our knowledge, this is the first study to jointly consider both (i) general aspects of restaurants (location, opening times and popularity) and (ii) specific item features (through users' preferences on specific types of food-drink deals), for recommending restaurant deals for residents and visitors in a city. Some services and apps, such as Wriggle [1], have recently arisen in which users in Bristol, Cardiff and Brighton can search for available restaurant deals in their area.

We present a model for recommending temporary deals offered by restaurants, taking account of (i) users' preferences on food-drink categories, (ii) contextual information and (iii) restaurant popularity. In our approach, the recommendable items are deals offered by restaurants, rather than restaurants "as a whole". We investigate the problem of weighting (balancing) and aggregating similarity information for the three aforesaid aspects. In addition, we conduct a case study and a preliminary evaluation with real user and restaurant deal data provided by Wriggle on three UK cities. The results hint that by setting the weighting parameters for balancing the aforesaid sources from user to user, our proposed scheme has the potential for addressing the cold start problem (e.g. first-time visitors to a city with no purchase history), hence becoming adaptable to both local residents and tourists.

2 MODEL

Let $u_i \in U$ be the ith user and U the set of all users. Denote by $C = \{c_1, \ldots, c_M\}$ the set of existing food-drink categories in the system, e.g. 'cocktails', 'tapas', 'Indian', 'Chinese', etc. Given M categories, every user u_i has associated a preference vector $P_i = (p_{i1}\, p_{i2} \ldots p_{iM})$ where $p_{ik} \in \{0, 1\}$ is a preference indicator towards category c_k by u_i. In our current version of the model, the value of p_{ik} is binary and determined depending on whether the user consumed deals under c_k or not. A restaurant deal $x_j \in X$, with X the set of all restaurant deals (item set), can have associated one or more categories $c_k \in C$. Thus, we formally define a deal as a tuple $x_j = \langle r_{x_j}, C_j, V_j \rangle$,

[1] Wriggle website: https://www.getawriggleon.com/

where r_{x_j} is the restaurant that offers the deal. C_j is the temporal context of the deal, namely start and end date at which the deal is available, and whether it is a lunchtime and/or dinner time deal. $V_j = (v_{j1}\ v_{j2} \ldots v_{jM})$ is a binary feature vector associated with the offer, in which $v_{jk} = 1$ if the deal x_j is labeled with category c_k, and $v_{jk} = 0$ otherwise. Our solution consists of two major stages: a context-based item pre-filtering stage, and a weighted filtering stage.

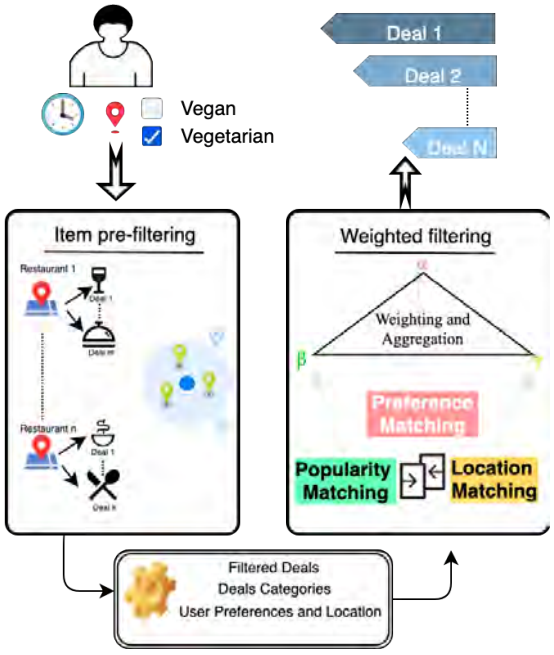

Figure 1: Architecture of the model for restaurant deal recommendation

Item pre-filtering. Unlike rating-based context pre-filtering approaches in the literature [5], in our model given an item set X (i.e. restaurant deals) context information C is firstly used to extract a subset of data related to the items relevant to that context. This is fundamental in domains where contextual limitations imply that not all existing items may be relevant or accessible by the end user at a certain place or time. For the scope of this study focused on the *Wriggle* data, we extract a subset of relevant deals to the current user and their context, accomplishing: (i) *Start-End Time*: most deals are periodical or limited and have a start-end time, therefore the currently available deals must be filtered; (ii) *Lunch or Dinner Time*: some deals are only active at lunchtime or dinner time, hence unavailable deals at a given time of the day are filtered out; and (iii) *Dietary Requirements*: although this is a user profile feature, we pre-filter suitable deals for users who are vegetarian or vegan.

Weighted filtering. This stage applies *three* matching processes and then weighs and aggregates resulting similarities:

(1) *Preference Matching*: It calculates the similarity between u_i preferences on food-drink categories, given by P_i, and the specific categorical features of a deal x_j, given by V_j. The cosine similarity is determined between both one-dimensional vectors, $m_\alpha(u_i, x_j) = sim(P_i, V_j)$. In essence, this filtering process entails a content-based approach relying on user preferences and item features of deals, hence it can easily integrate other content-based models in extant literature.
(2) *Popularity Matching*: This process takes the restaurant popularity into account, based on the average customer rating given to the restaurant. The popularity matching is calculated as the average customer rating of the restaurant r_{x_j}, thus $m_\beta(u_i, x_j) = pop(r_{x_j})$. Despite its simplicity, this solution is not personalised for the end user in question, because it is only dependent on r_{x_j}. An alternative personalised solution would be to apply a Collaborative Filtering (CF) algorithm to identify the K most similar users to u_i who rated r_{x_j}, based on their preference vectors P_i, and predicting how popular the restaurant might be for u_i.
(3) *Location Matching*: It takes the distance between restaurants within a predefined radius and the current user location, thereby prioritising deals from closer restaurants:

$$m_\gamma(u_i, x_j) = 1 - \frac{dist(u_i, r_{x_j})}{radius} \quad (1)$$

One of the contributions in this study is an adaptive weighting scheme for balancing preferences, popularity and location. Let α, β and γ be the weighting parameters or degrees of influence played by the preference, popularity and location matching, respectively. Without loss of generality, $\alpha, \beta, \gamma \in [0, 1]$ and $\alpha + \beta + \gamma = 1$. The overall matching used for selecting and recommending the top-N deals for u_i, is:

$$m(u_i, x_j) = \alpha \cdot m_\alpha(u_i, x_j) + \beta \cdot m_\beta(u_i, x_j) + \gamma \cdot m_\gamma(u_i, x_j) \quad (2)$$

We now describe a preliminary solution for adaptively setting α, β and γ for every user. It is worth noting that deeper investigation of applying more advanced optimisation or machine learning techniques to optimally set these weights, constitutes our immediate future work.

The influence of α, which refers to the user preferences on food-drink types, should rely on the size of the user's purchase history, i.e. the number of deals previously consumed. Users with a longer history have more accurately built preferences P_i than (cold) users with a short history, hence α should be higher in the former case. For users with no purchase history, e.g. first-time visitors to a city, for instance), preference information in P_i should be disregarded by setting $\alpha = 0$. Inspired by fuzzy set theory, we achieve this by setting $\alpha \in [0, \alpha_{max}]$, $0 < \alpha_{max} < 1$, such that α increases as the user history grows.

The influence of β relies on the amount of ratings received by the restaurant associated to x_j. If r_{x_j} has more customer ratings, β should be higher under the premise that frequently rated restaurants have more reliable (less biased) popularity information, and vice versa. Likewise, for a new restaurant with no ratings, we set $\beta = 0$. Using a similar principle as the one for α, we set $\beta \in [0, \beta_{max}]$, $0 < \beta_{max} < 1 - \alpha_{max}$.

The influence of γ, which refers to the proximity between user and venue, is (without losing generality) determined upon the other two parameters, as $\gamma = 1 - (\alpha + \beta)$. In other words, distance becomes more relevant if u_i has a smaller purchase history or r_{x_j} has less customer ratings. If both α and $\beta = 0$, the filtering process between a cold user and a deal offered by an unrated restaurant becomes purely location-based, $\gamma = 1$.

3 EXPERIMENTAL CASE STUDY

This section presents a case study conducted in collaboration with Wriggle, on a real dataset describing restaurants, deals and purchases made by users who used Wriggle in Bristol, Cardiff and Brighton. By using the purchase history and user profile, a sensitivity analysis is conducted on our proposed model parameters.

Dataset Description. The anonymised datasets provided by Wriggle contain a history of purchased deals by every user over a period of five years, between 2014 and 2019. Around 305K purchases are logged by 141K users. Also, a total of approximately 11K deals offered by 2153 restaurants are included in the dataset, with each deal being associated to one or multiple categories, out of a total of 63 categories describing food or drink characteristics/cuisines. There is also data about every user's profile, including dietary requirements if any (vegetarian, vegan), and restaurant profiles that contain the restaurant's average popularity based on users' rating on deals offered by that restaurant.

Experimental Setting. We filter users who have at least one purchase in the last 5 months of purchase dataset because real location data exists only for that particular period. Then, we split the user history dataset into a training and test set for three major cities, Bristol, Cardiff and Brighton, that Wriggle operates currently. We consider three different time span settings for the user purchase history: 6 Months, 12 Months and entire history since 2014. We then separate the latest deal with location information purchased by each user into the test set. Users with three or less items in their purchased history have been removed for the purpose of this experiment, leaving a consolidated purchase history of 2043 Users for Bristol, 249 for Cardiff and 643 for Brighton. Category information retrieved from deals in the purchase history is used to built preference vector of user P_i for the preference matching. Likewise, the information about restaurant popularity, opening times and location are retrieved from the restaurant-related data. For the contextual information, location data and time are inferred by retrieving the temporal information associated to the last purchased deal (test data). Finally, we consider $k = 10$ for the size of the recommendation list.

Evaluation Metrics. We recommend the top-k matching offers to the target user and investigate the predictive power exhibited by the model in recommending the (removed) latest deal purchased by each user, or the restaurant which offered it. For this end, the performance evaluation metrics employed are adapted versions of average recall@k and average NDCG@k on all users, thereby predicting the appearance of each user's latest deal or visited restaurant in her history in the recommendation list. The average recall is:

$$avg\_recall@k = \frac{\sum_{u_i \in U} y_i}{|U|} \tag{3}$$

$$y_i = \begin{cases} 1 & \text{if last deal consumer by } u_i \text{ is among top-}k, \\ \frac{1}{2} & \text{if last restaurant visited by } u_i \text{ is among top-}k, \\ 0 & \text{otherwise.} \end{cases}$$

Average Normalised Discounted Cumulative Gain at k:

$$avg\_NDCG@k = \frac{\sum_{u_i \in U} NDCG@k_i}{|U|}$$

$$NDCG@k_i = \sum_{j=1}^{k} \frac{2^{z_{i,j}} - 1}{log_2(j+1)} \tag{4}$$

where $z_{i,j} = 1$ if the last deal consumer by u_i is the jth recommended item, $z_{i,j} = 0.5$ if the restaurant last visited by u_i is at the jth recommended item, and $z_{i,j} = 0$ otherwise.

Results and Discussion. Three baseline approaches, and two versions of the proposed model with non-null weights, are considered:

Most Popular: Recommend deals based on venue popularity.
User-Preference: Recommend deals predicted on preferences over categories in deals.
Location: Recommend deals based on restaurant proximity.
Same Weight: Popularity, preferences and context are equally important for every user and restaurant, i.e. $\alpha = \beta = \gamma$.
Optimised Weight: It adaptively sets weights as explained in Section 3, with $\alpha_{max} = \beta_{max} = 0.3$. Both α (resp. β) become maximum when the user history length (resp. restaurant rating count) is greater than five.

Figure 2 summarises the average results obtained by the five models, for users in the three cities considered and the three time span settings considered. Despite a more exhaustive validation is needed, the results provide some interesting insights.

The proposed model with optimised weight scheme tends to slightly outperform the version with same weights, in almost all cases, specially when considering a shorter time span (6 months). Whilst this improvement is not significant,

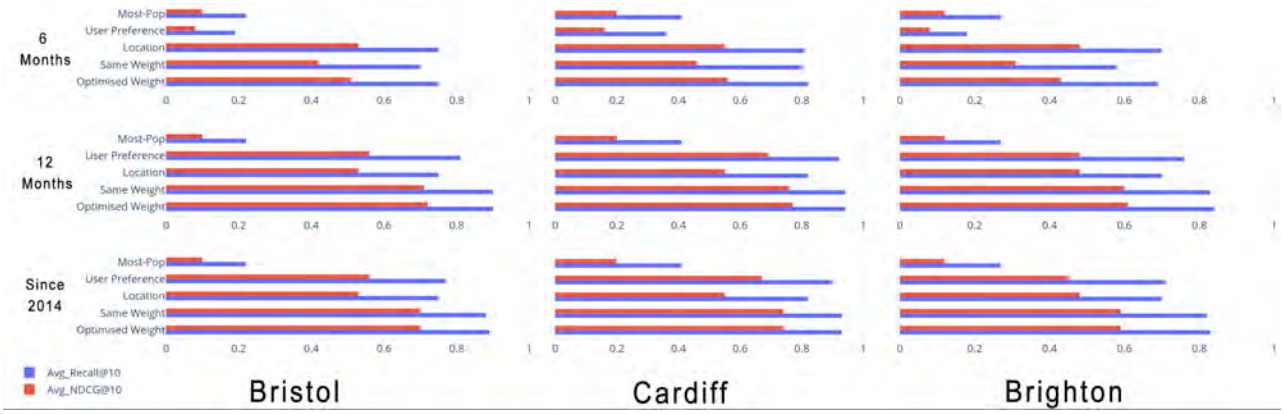

Figure 2: Comparison results in terms of average recall and average NCDG for $k = 10$

it motivates us to investigate how to further improve it by devising more user-adaptive weight optimisation methods in future work. Both two versions of our model generally outperform the three baseline approaches, however a location based recommendation has better predictive power in two of the three cities for the 6-month case. This suggests that most users may have a scarce purchase history in such a short time span, in which case prioritising restaurant proximity might increase the chances for better predictions.

Finally, the fact that the user preference baseline gently improves for longer time spans, suggests that the more purchase history data are available, the more reliable the extracted (implicit) preference information is.

4 CONCLUSION

This contribution proposes a recommendation model for suggesting restaurant deals to local and visiting users to a city by balancing their food-drink preferences, the popularity of the restaurant, and the context surrounding the user, such as his/her location. A case study has been conducted with real data provided by Wriggle, with insightful results motivating the need for follow-up research on how to optimally balance multiple information sources.

People often visit restaurants in groups whose members have diverse preferences. Accordingly, future work involves investigating preference aggregation for consensual group recommendations [12, 13]. We are also interested in (i) harnessing the capabilities of data networks in smart cities to enable highly situation-aware recommendations in real time, specially for tourists visiting a city; (ii) modeling users' preferences on food-drink categories more flexibly and under several decision criteria; and (iii) applying improved models on open datasets to make this research more reproducible.

5 ACKNOWLEDGMENTS

The authors would like to thank Rob Hall (CEO, Wriggle) and Clement Debiaune (CTO, Wriggle) for incentivising and fostering the collaboration that made this research possible.

REFERENCES

[1] D. Herzog, C. Laβ, Wolfgang Wörndl. Tourrec: a tourist trip recommender system for individuals and groups. Proceedings 12th ACM Conference on Recommender Systems (Recsys'18), pp. 496-497, 2018.
[2] A. Moreno, A. Valls, D. Isern, L. Marin, J. Borrás. SigTur/E-Destination: Ontology-based personalized recommendation of Tourism and Leisure Activities. Engineering Applications of Artificial Intelligence,
[3] E. Ezin, I. Palomares, J. Neve. Group Decision Making with Collaborative-Filtering ín the loop: interaction-based preference and trust elicitation. Accepted, IEEE SMC 2019 Conference. In press.
[4] M. Al-Ghossein, T. Abdessalem, A. Barré. Cross-Domain Recommendation in the Hotel Sector. Proceedings RecTour'18, in 11th Conf. ACM Recsys'18, pp. 1-6, 2018.
[5] G. Adomavicius, A. Tuzhilin. Context-Aware Recommender Systems. In F. Ricci et al. (Eds.) Recommender Systems Handbook, pp. 217-253, Springer, 2011.
[6] A. Ebadi, A. KrzyÅijak. A Hybrid Multi-Criteria Hotel Recommender System Using Explicit and Implicit Feedbacks. International Journal of Computer and Information Engineering, 10(8), 1450-1458, 2016.
[7] P. Longart. Consumer Decision Making in Restaurant Selection. PhD Thesis, Buckinghamshire New University, 2015.
[8] L. Li, Y. Zhou, H. Xiong, C. Hu, X. Wei. Collaborative Filtering based on User Attributes and User Ratings for Restaurant Recommendation. Proceedings 2nd IEEE IAEAC Conference, pp. 2592-2597, 2017. 26(1), pp. 633-651, 2013.
[9] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis. Predicting Your Next Stop-over from Location-based Social
[10] J. Zeng, F. Li, H. Liu, J. Wen, S. Hirowaka. A Restaurant Recommender System based on User and Location in Mobile Environment. Proceedings 5th IIAI International Congress, pp 55-60, 2016. Network Data with Recurrent Neural Networks. Proceedings RecTour'17, 11th Conf. ACM Recsys'17, pp. 1-8, 2017.
[11] M. Bressan, S. Leucci, A. Panconesi, P. Raghavan, E. Terolli. The Limits of Popularity-Based Recommendations, and the Role of Social Ties. Proceedings 22nd ACM SIGKDD International Conference, pp. 745-754, 2016.
[12] A. Delic, J. Neidhardt, H. Werthner. Group Decision Making and Group Recommendations. Proceedings 20th IEEE CBI, pp. 79-88, 2018.
[13] I. Palomares. Large Group Decision Making: creating Decision Support Systems at Scale. Springerbriefs in Computer Science, Springer, 2018.

Re-CoSKQ: Towards POIs Recommendation Using Collective Spatial Keyword Queries

Ramon Hermoso*
rhermoso@unizar.es
Department of Computer Science and Systems Engineering, University of Zaragoza
Zaragoza, Spain

Sergio Ilarri
silarri@unizar.es
Department of Computer Science and Systems Engineering, I3A, University of Zaragoza
Zaragoza, Spain

Raquel Trillo-Lado
raqueltl@unizar.es
Department of Computer Science and Systems Engineering, I3A, University of Zaragoza
Zaragoza, Spain

ABSTRACT

The goal of collective spatial keyword queries is to retrieve, from a spatial database, a group of spatial items such that the description of the items included in that set (typically based on the use of keywords) is completely covered by the query's keywords. Moreover, it ensures that the items retrieved are as near as possible to the query location and have the lowest inter-item distances. We argue that using this concept in the field of recommender systems could be useful. Therefore, in this position paper, we outline the idea of Re-CoSKQ, an adaptation of Collective Spatial Keyword Query (CoSKQ) for recommender systems in the tourism domain to provide the user with a set of Points of Interest (POIs) that satisfy his/her queries both geographically and semantically.

CCS CONCEPTS

• **Retrieval tasks and goals → Recommender systems**.

KEYWORDS

Collective spatial keyword querying, recommender systems, tourism

1 INTRODUCTION

Recommender systems (RS) have been studied for several decades, aiming to facilitate item selection as part of the user's decision-making processes [11]. One of the hard challenges of recommender systems is to provide successful responses to user queries, especially when little information is available. In most RS approaches, algebraic operations with user-item rating matrices allow predicting the future likeness of new items for a user (e.g., using collaborative filtering, content-based, or hybrid approaches). However, when the suitability of the suggested items depends on different features such as the location of items and users, textual descriptions of items, or the (sometimes blurry) query description, those approaches face new problems to address. For example, for the recommendation of points of interest (POIs), the location of the items and the user, as well as other context attributes, may play a key role [6].

The idea of Collective Spatial Keyword Querying (CoSKQ) emerged some years ago as a promising technique to query spatial databases containing information about items and their location [2]. It puts forward a smart solution to retrieve a group of spatial items such that the description of the items included in that set (typically based on the use of keywords) is completely covered by the query's keywords and assures that the items are as near as possible to the query location and have the lowest inter-item distances.

*All authors contributed equally to this research.

We believe that exploiting spatial keyword querying as a basis to build recommender systems is an interesting research avenue to explore. Therefore, combining both fields of research, in this position paper we present the idea of Re-CoSKQ, a recommender system that uses CoSKQ to provide a set of items that semantically covers the keywords of a query (even if they do not match perfectly) and minimizes the cost, in terms of the distance to get to them and the similarity between query keywords and item descriptions.

As a problem statement, let us consider a set $U = \{u_1, ..., u_n\}$ of users spending their time in a city as tourists. Let $\mathbb{O} = \{o_1, ..., o_m\}$ be a set of POIs, i.e., spots with some kind of relevant attraction for visitors. Examples of POIs could be museums, monuments, parks, or buildings with some historical flavour, just to mention a few. Now, let $o_i.\kappa = \{k_1, ..., k_j\}$ be a set of keywords with which a POI $o_i \in \mathbb{O}$ is described. These keywords can usually be retrieved in an automated way by using semantically-annotated resources. Moreover, every POI $o_i \in \mathbb{O}$ is placed in a location denoted by $o_i.\lambda$.

Re-CoSKQ uses collective spatial keyword querying in order to cope with the location of POIs and users and also with the similarity between the keywords in the user's query and the description of the POIs. Let $q = \langle \lambda, \kappa \rangle$ be a user's query, where $q.\lambda$ represents the user's location and $q.\kappa$ stands for the query split in keywords (only relevant words for the search are taken into account). The main goal is to provide a method to return a set of items $\mathbb{O}' \subseteq \mathbb{O}$ which semantically covers the keywords in $q.\kappa$ and also ensures that their cost, in terms of distance –between the POIs and the user who issued the query– and the similarity of terms, is minimal.

The next sections intend to shed some light into the problem and present the approach we have envisioned to deal with it. Specifically, the rest of the paper is structured as follows. First, Section 2 we revise the concept of CoSKQ. Then, in Section 3, we present the Re-CoSKQ approach. In Section 4, we sketch an evaluation proposal. Finally, in Section 5, we conclude with a summary and some future work.

2 BACKGROUND: CoSKQ

As we have previously stated, CoSKQ attempts to find the solution to the problem of retrieving a group of spatial objects that collectively match the user preferences given specific locations (of the user and also of the objects) and a set of keywords. The method is designed to work with spatial databases, so it does much effort on providing an efficient computation, in terms of the data structure used and how data are accessed [2, 3]. Although going in depth on the subtle considerations of the method is out of the scope of this paper, we summarize how it works applied to our domain.

It uses the concept of IR-tree data structures [4] to efficiently store information about POIs. This type of structure allows indexing objects and the keywords which describe them as well as their spatial position. IR-trees are a type of balanced trees in which each leaf node contains an item o (a POI object), a bounding rectangle of o and an item identifier, while each non-leaf node in the tree contains a pointer to a child node, a *Minimum Bounding Rectangle* (MBR) of all rectangles in entries of the child node, and an item identifier containing the set of all keywords in the entries of the child node. Moreover, each leaf node contains a pointer to an inverted file with the keywords that describe the POIs stored in that node. Figure 1 depicts an example for which CoSKQ may offer a solution with a query q and a set of POIs $\{o_1, ..., o_{10}\}$. Figures 2 and 3 show how the data are geographically partitioned and stored in an IR-tree.

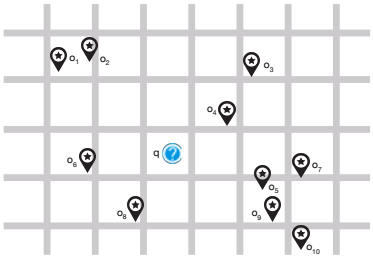

Figure 1: Example of a possible scenario

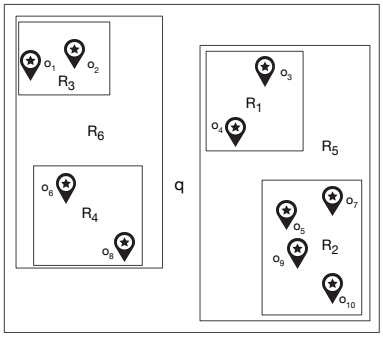

Figure 2: Item positioning for the example

CoSKQ presents different algorithmic solutions based on minimizing a cost function. The chosen cost function may vary depending on the authors of each specific proposal and the scenario where it is applied. Different cost functions, taking into account the distances between items and query locations, can be found in [2, 3]. It has been proved that solving a spatial group keyword query is an NP-complete problem [2], i.e., the performance of an exact algorithm does not present itself as a reasonable solution, in terms of running time and I/O cost [7]. For that reason, some approximation algorithms have been developed to calculate the output sets of objects [2, 3, 7, 12]. Besides, in special cases, the application of an exact algorithm may be plausible, especially when the number of keywords in the query is small. Some exact algorithms, based on dynamic programming for minimizing the cost function are presented in [2, 3, 7].

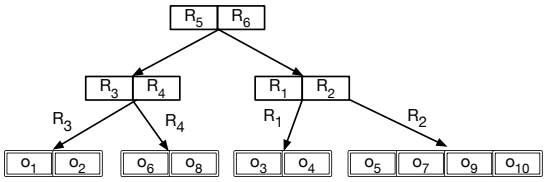

Figure 3: Resulting IR-tree containing data for the example

3 Re-CoSKQ APPROACH

We present Re-CoSKQ as an instantiation of the CoSKQ problem, especially designed for recommendations in the tourism domain (i.e., the user is a tourist and the items are points of interest that the user may want to visit). The most common instantiation of CoSKQ assumes that the set of keywords describing the POIs in the query result must contain, at least, all the keywords contained in the query [2]. Formally, $q.\kappa \subseteq \cup_{o'_i \in \mathbb{O}'} o'_i.\kappa$, where $\mathbb{O}'$ is the set of POIs calculated as a result of a user issuing the query q; for simplicity, from now on, we will use $o \in \mathbb{O}'$ to avoid o'_i. However, there are scenarios in which this assumption must hold some more hard constraints. For instance, when tackling a recommendation problem, we need to ensure not only that the keyword query is covered by the resulting $\mathbb{O}'$ but also that both the maximum distance between the query location and any of the POIs in $\mathbb{O}'$ and the maximum distance between any two POIs in $\mathbb{O}'$ are minimized.

Moreover, in this paper, we do not assume that $q.\kappa$ can be fully covered. Actually, we claim that this assumption may derive in empty sets in many recommendation scenarios where queries are expressed, for instance, with different vocabularies, or where they cannot be easily solved with the given descriptions of POIs. Thus, we believe that it is important to provide query outcomes even when full keyword coverage is not possible. In order to do that, we propose to use a similarity function to calculate how similar the keywords in $q.\kappa$ are compared to those in $\cup_{o \in \mathbb{O}'}$. For example, given $q.\kappa = \{outdoors, animals, kids\}$, if located nearby, one of the POIs included in the outcome could be a zoo, which could be described by a set of keywords $\{open\text{-}air, birds, snakes, mammals, family\}$. This object would never be returned using a classic CoSKQ approach, but considering the semantic similarity between keywords one can easily observe that the terms are related, since *birds*, *snakes* and *mammals* are types of *animals*, *outdoors* and *open-air* are synonyms and *kids* are part of *families*. We will present how to cope with this when presenting different cost functions.

3.1 Cost Analysis

Re-CoSKQ attempts to minimize the cost of finding an appropriate set of POIs for a given query q. This cost is modelled as a function that depends on distances between the query and the locations of POIs as well as between the keywords. Different equations have been proposed to model cost in the CoSKQ problem [1, 2, 10]. In the following, we redefine some of them for the Re-CoSKQ problem.

TYPE 1. A linear combination of the maximum distance between the query location and any POI in $\mathbb{O}'$, the maximum pairwise distance between any two POIs in $\mathbb{O}'$, and the maximum of the semantic distance between the query keywords ($q.\kappa$) and the set of

keywords in $\mathbb{O}'$, i.e. $\cup_{o\in\mathbb{O}'} o.\kappa$. It is formally defined in Eq. 1.

$$cost(q,\mathbb{O}') = \alpha \cdot \max_{o\in\mathbb{O}'}[dist(q.\lambda, o.\lambda)] + \beta \cdot \max_{o_1,o_2\in\mathbb{O}'}[dist(o_1,o_2)] + \omega \cdot \max_{k_1\in q.\kappa, k_2\in\cup_{o\in\mathbb{O}'} o.\kappa}[dist(k_1,k_2)] \quad (1)$$

where $\alpha+\beta+\omega=1$ are weights to denote the relevance of each of the three types of distances involved, which allow adding up distances which may have different ranges of values.

TYPE 2. This type of function defines cost as the maximum of the three factors in the TYPE 1 function; i.e., the highest value between the maximum distance among the query location and any POI in $\mathbb{O}'$, the maximum pairwise distance between any two POIs in $\mathbb{O}'$, and the maximum of the semantic distance between query keywords ($q.\kappa$) and the set of keywords in $\cup_{o\in\mathbb{O}'} o.\kappa$. This is formally defined by Eq. 2, where again weights α, β and ω are used.

$$cost(q,\mathbb{O}') = \max\left\{\alpha \cdot \max_{o\in\mathbb{O}'}[dist(q.\lambda, o.\lambda)], \beta \cdot \max_{o_1,o_2\in\mathbb{O}'}[dist(o_1,o_2)], \omega \cdot \max_{k_1\in q.\kappa, k_2\in\cup_{o\in\mathbb{O}'} o.\kappa}[dist(k_1,k_2)]\right\} \quad (2)$$

TYPE 3. This function uses a min-max approach, linearly combining the minimum distance between the query location and any POI with the maximum values for pairwise distance between any two POIs in $\mathbb{O}'$ and the semantic distance between query keywords ($q.\kappa$) and the set of keywords in $\mathbb{O}'$, i.e., $\cup_{o\in\mathbb{O}'} o.\kappa$ (see Eq. 3).

$$cost(q,\mathbb{O}') = \alpha \cdot \min_{o\in\mathbb{O}'}[dist(q.\lambda, o.\lambda)] + \beta \cdot \max_{o_1,o_2\in\mathbb{O}'}[dist(o_1,o_2)] + \omega \cdot \max_{k_1\in q.\kappa, k_2\in\cup_{o\in\mathbb{O}'} o.\kappa}[dist(k_1,k_2)] \quad (3)$$

again with $\alpha+\beta+\omega=1$.

TYPE 4. This is a unified cost function, adapted from [3], that generalizes types 1 to 3 in one function. It is presented in Eq. 4.

$$cost(q,\mathbb{O}') = \left[\left(\alpha \cdot \Big(\sum_{o\in\mathbb{O}'}(dist(q.\lambda, o.\lambda))^{\phi_1}\Big)^{\frac{1}{\phi_1}}\right)^{\phi_2} + \left(\beta \cdot \max_{o_1,o_2\in\mathbb{O}'} dist(o_1,o_2)\right)^{\phi_2} + \left(\omega \cdot \max_{k_1\in q.\kappa, k_2\in\cup_{o\in\mathbb{O}'} o.\kappa} dist(k_1,k_2)\right)^{\phi_2}\right]^{\frac{1}{\phi_2}} \quad (4)$$

with $\alpha+\beta+\omega=1$, $\phi_1\in\{-\infty,1,\infty\}$ and $\phi_2\in\{1,\infty\}$. The ϕ_1 and ϕ_2 values stand for tuning parameters, allowing to describe the previous cost functions (types 1-3) by varying their values. For example, an instantiation with $\alpha,\beta,\omega=\frac{1}{3}$, $\phi_1=\infty$ and $\phi_2=1$ results in a Type 1 cost function with the weights α, β, and ω indicated:

$$cost(q,\mathbb{O}') = \frac{1}{3}\left(\max\sum_{o\in\mathbb{O}'}(dist(q.\lambda, o.\lambda)) + \max_{o_1,o_2\in\mathbb{O}'} dist(o_1,o_2) + \max_{k_1\in q.\kappa, k_2\in\cup_{o\in\mathbb{O}'} o.\kappa} dist(k_1,k_2)\right)$$

3.2 Distance Analysis

As we have pointed out, there exist different distance functions needed to calculate the cost in Re-CoSKQ. Analyzing any of the proposed cost functions, we can observe that there are three different distance instantiations, as we explain in the following.

Location distance. ($dist(q.\lambda, o.\lambda)$) refers to the physical distance between the query location and a POI's location. It can be calculated with different geometrical approaches. In the following, we point out some possible functions.

Euclidean distance. It is probably the most common distance function used in the literature for many different types of problems and domains. It is formally defined by Eq. 5:

$$dist(q.\lambda, o.\lambda) = \sqrt{\sum_{i=1}^{n}(q.\lambda_i - o.\lambda_i)^2} \quad (5)$$

We assume that the position of queries and POIs are given by a pair of coordinates $\langle lat, long\rangle$. This distance may work well when the routes between POIs are roughly calculated or the users can walk straight from any location to another.

$\mathcal{L}_1$-Norm. It is another well-known distance function, also known as *Manhattan distance*. It calculates the sum of the magnitudes of the vectors in a space, i.e., the sum of absolute difference of the components of the vectors (see Eq. 6).

$$dist(q.\lambda, o.\lambda) = \sum_{i=1}^{n}|q.\lambda_i - o.\lambda_i| \quad (6)$$

We use 2-dimension spaces, denoted by location coordinates. This distance may be suitable for grid-based scenarios, e.g., POIs in a city connected by roads/paths, or halls in a museum linked by corridors.

Geodesic distance. It is the type of function we need if we use a graph to model how POIs and users are connected. The geodesic distance is defined as the shortest path between two vertices in a graph. This is useful when modeling a scenario with weighted edges, since some extra information can be added (e.g., about congested routes or crowded halls). Many algorithms can be used to calculate shortest paths in graphs (e.g., the Dijkstra's algorithm).

POI-to-POI distance. ($dist(o_1,o_2)$) could also be called intra-POI distance, since it calculates the distance between two POIs. Note again that the location of $o\in\mathbb{O}'$ is denoted by $o.\lambda$. As we assume a 2-dimension space in Re-CoSKQ, we can reduce the calculation of this distance to the problem of calculating the location distance. Thus, the same functions described above may apply to this case.

Term distance. ($dist(k_1,k_2)$) is the distance we use to calculate how similar two different keywords are. In this case, we compare the query keywords ($q.\kappa$) and the keywords in $\cup_{o\in\mathbb{O}'} o.\kappa$. In the cost function, we try to minimize the maximum distance between the $q.\kappa$ set and $o.\kappa$ in a pairwise basis. In order to calculate the similarity between keywords, we adhere to ontology-based measures, typically used in semantic web approaches. This type of measures usually calculates the similarity according to structured knowledge defined by an ontology. In the following, we propose some functions that we consider to be suitable for the Re-CoSKQ problem; once the similarity has been estimated, we should provide a way to calculate the distance associated to it, such as $dist(k_1,k_2)=1-sim(k_1,k_2)$.

Similarity based on concept closeness. This measure takes into account the closeness of the concepts in the hierarchical tree representing the ontology. It is based on the *relatedness* property presented in [8] and is defined as $sim(k_1,k2)=1-\frac{sp(k_1,k_2)}{2D}$, where $sp(\cdot)$ is a function that returns the shortest path between the two terms in the ontology tree and $2D$ denotes the maximum distance between any two concepts in the ontology (D is the ontology depth).

Similarity based on closeness and concept depth. This measure, proposed in [9], takes into account the closeness of keywords in

the ontology but also the depth in the ontology tree where they can be found (see Eq. 7). It assumes that the semantics of concepts are more general in higher levels. Thus, the higher we find the concept in the ontology the lower the similarity while, on the contrary, the lower we find the concepts the higher the similarity.

$$sim(k_1, k_2) = \begin{cases} e^{-\alpha l} \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if \quad k_1 \neq k_2 \\ 1 & otherwise \end{cases} \tag{7}$$

where l is the shortest path between k_1 and k_2 and h is the depth of the least common subsumer of both concepts. Parameters $\alpha, \beta > 0$ are weights to modulate the contribution of these factors.

3.3 Outline of Processing Issues for Re-CoSKQ

Several algorithms address the problem of implementing CoSKQ. CoSKQ is an NP-complete problem, so exact algorithms (e.g., the linear programming approaches presented in [2, 3]) only make sense when the number of query keywords is low. However, on average conditions, an approximate algorithm is needed. Different approaches using greedy techniques and pruning steps have been presented to reduce the needed resources. Due to lack of space, we omit further details and refer the reader to [2, 3] for further revision on approximate algorithms for CoSKQ. Re-CoSKQ needs to tackle an optimization problem to try to minimize the cost function.

4 EVALUATION PROPOSAL

Most works on CoSKQ focus their evaluation on measuring the performance (in terms of running time) and approximation ratio. Nevertheless, when applying this approach to recommender systems we are not only interested in these issues, as the quality of the recommendation is also key. An interesting problem is that full coverage is assumed in classic CoSKQ; that is, $\cup_{o \in \mathbb{O}'} o.\kappa$ is assumed to contain, at least, all keywords in $q.\kappa$. This is not the case of Re-CoSKQ, where the coverage is estimated by keyword similarity. Moreover, the evaluation usually needs a ground truth to compare with, in order to be able to calculate accuracy metrics such as precision and recall. As far as we know, there is no dataset annotated with this type of information. Thus, we propose to first define a set of interesting and representative keyword queries and then manually annotate a dataset containing POIs descriptions with the keywords by assigning each POI to a set of predefined categories (much smaller than the number of keywords) defined based on the queries that have been selected for evaluation, in order to define a dataset with information that can represent a suitable ground truth to compare with. Precision and recall may be calculated by comparing the retrieved POIs according to the categories specified by the user in the query. Regarding the items, there are many datasets that contain information about geographic locations and keyword descriptions; a tailored synthetic dataset could also be generated by using DataGenCARS [5]. All this could be complemented with a user-centered evaluation.

The main idea in the empirical evaluation is to show the benefits of the proposal and test how different cost functions behave, tuning different parameters. We are also interested in the scalability of the proposal, so tests with different numbers of query keywords and POIs (as well as simultaneous queries/users) should be carried out. Moreover, in order to check the feasibility concerning the use of resources, we will use exact and approximate algorithms to test their performance (running time and approximation ratio).

5 CONCLUSIONS AND FUTURE WORK

In this position paper, we have presented the idea of Re-CoSKQ, a collective spatial keyword query approach for recommender systems, where keyword coverage is not assumed, by considering keyword similarity. We have tackled the problem as a minimization problem, for which we have defined some cost functions.

We are currently working on an empirical evaluation to test the approach and its benefits over other POI recommendation approaches. Furthermore, we would like to extend the approach to group recommendation; that is, different users in different locations will issue their queries and the opportunity of group visits (groups of people visiting the same items together) will be explored, so the problem becomes more complex, since $\mathbb{O}'$ must contain suitable POIs that satisfy all users (or at least a set of them). We are also interested in dynamic environments where both the POIs and users could potentially be on the move and context conditions can change quickly over time. Finally, we also intend to consider other spatial distance calculation approaches, such as heuristic searches (e.g., by using $A^\star$ algorithms), as well as other approaches to compute term distances (e.g., a word embedding approach such as word2vec).

Acknowledgments

Work supported by the project TIN2016-78011-C4-3-R (AEI/FEDER, UE) and the Government of Aragon (Group Reference T35_17D, COSMOS group) and co-funded with Feder 2014-2020 “Construyendo Europa desde Aragon”.

REFERENCES

[1] Xin Cao, Gao Cong, Tao Guo, Christian S Jensen, and Beng Chin Ooi. 2015. Efficient processing of spatial group keyword queries. *ACM Transactions on Database Systems (TODS)* 40, 2 (2015), 13:1–13:48.
[2] Xin Cao, Gao Cong, Christian S Jensen, and Beng Chin Ooi. 2011. Collective spatial keyword querying. In *2011 ACM SIGMOD Int. Conference on Management of Data*. ACM, 373–384.
[3] Harry Kai-Ho Chan, Cheng Long, and Raymond Chi-Wing Wong. 2018. On generalizing collective spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1712–1726.
[4] Gao Cong, Christian S Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment* 2, 1 (2009), 337–348.
[5] María del Carmen Rodríguez-Hernández, Sergio Ilarri, Ramón Hermoso, and Raquel Trillo-Lado. 2017. DataGenCARS: A Generator of Synthetic Data for the Evaluation of Context-Aware Recommendation Systems. *Pervasive and Mobile Computing* 38, Part 2 (2017), 516–541.
[6] María del Carmen Rodríguez-Hernández, Sergio Ilarri, Raquel Trillo-Lado, and Ramón Hermoso. 2015. Location-Aware Recommendation Systems: Where We Are and Where We Recommend to Go. In *Int. Workshop on Location-Aware Recommendations (LocalRec)*, Vol. 1405. CEUR Workshop Proceedings, 1–8.
[7] Yunjun Gao, Jingwen Zhao, Baihua Zheng, and Gang Chen. 2015. Efficient collective spatial keyword query processing on road networks. *IEEE Transactions on Intelligent Transportation Systems* 17, 2 (2015), 469–480.
[8] Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database* 49, 2 (1998), 265–283.
[9] Yuhua Li, Zuhair A Bandar, and David McLean. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering* 15, 4 (2003), 871–882.
[10] Cheng Long, Raymond Chi-Wing Wong, Ke Wang, and Ada Wai-Chee Fu. 2013. Collective spatial keyword queries: a distance owner-driven approach. In *2013 ACM SIGMOD Int. Conference on Management of Data*. ACM, 689–700.
[11] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer.
[12] Pengfei Zhang, Huaizhong Lin, Bin Yao, and Dongming Lu. 2017. Level-aware collective spatial keyword queries. *Information Sciences* 378 (2017), 194–214.