Predicting Your Next Trip: A Knowledge Graph-Based Multi-task Learning Approach for Travel Destination Recommendation

AMINE DADOUN, Eurecom, France
RAPHAËL TRONCY, Eurecom, France
MICHAEL DEFOIN-PLATEL, Amadeus, France
GERARDO AYALA SOLANO, Amadeus, France

Recommending the next destination to a traveler is a task that has been at the forefront of the airline industry for a long time, and its relevance has never been more important than today to revive tourism after the Covid-19 crisis. Several factors influence a user's decision when faced with a variety of travel destination choices: geographic context, best time to go, personal experiences, places to visit, scheduled events, etc. The challenge of recommending the right travel destination lies in efficiently integrating and leveraging all of this information into the recommender system. Based on a real world application scenario, we propose a multi-task learning model based on a neural network architecture that leverages knowledge graph to recommend the next destination to a traveler. We experimentally evaluated our proposed approach by comparing it against the currently in-production system and state-of-the-art travel destination recommendation algorithms in an offline setting. The results confirm the significant contribution of using knowledge graphs as a means of representing the heterogeneous information used for the recommendation task, as well as the benefit of using a multi-task learning model in terms of recommendation performance and training time.

1 INTRODUCTION

With the digital transformation of airline sales points from physical to virtual stores, recommender systems have proven their value in facilitating the search and the decision making process for travelers faced with an increasingly wide selection of airline products [7]. However, current airline solutions which provide recommendations on travel destinations lack contextualization and, more importantly, personalization [7]. They either use a solution that suggests their most popular destinations to all travelers, or an interactive inspiration tool that matches travelers' criteria (budget, interests, etc.) with travel destinations. In our study, we focus on a passive inspiration scenario, in which travel destinations are sent to travelers through airline email marketing campaigns, with the aim of facilitating the search process for travelers.

The use of *collaborative filtering* (CF) methods for travel destination recommendation suffers from the cold start problem and data sparsity. Indeed, using only travelers' historical bookings as input information of the recommender system is generally not sufficient [8]. We hypothetize that incorporating additional information such as travel context, traveler demographics, or destination metadata into the recommender system will be valuable in addressing the above-mentioned issues. To integrate these heterogeneous information into a common data structure, we consider the development of knowledge graph as an appropriate and effective candidate method. Indeed, recent works [22, 24, 31] have demonstrated the effectiveness of using knowledge graph embeddings for items recommendation. However, as pointed out in [10], not all knowledge graph embedding algorithms are effective in combining different types of literals and most of them do not have a proper mechanism to handle multi-valued literals (text, image, numerical value, etc.). Inspired by the work proposed in [32], where the authors propose an approach for both relational learning and non-discrete attribute prediction on knowledge graphs, we propose **K**nowledge **G**raph-based **M**ulti **Task** **L**earning

For **Rec**ommendation (KGMTL4Rec[1]), a neural network-based multi-task learning algorithm for travel destination recommendation that leverages knowledge graph information[2]. The model architecture is depicted in Figure 3.

Our contributions can be summarized as follows: (a) we build a knowledge graph encompassing both travelers' historical bookings (collaborative information) as well as booking contexts, travelers' and destinations' metadata from Linked Open Data; (b) we propose a multi-task learning model to learn vector representations of knowledge graph entities that we named KGMTL4Rec, and we use this model to compute travel destination recommendation scores between travelers and destinations; (c) we conduct extensive experiments to compare KGMTL4Rec with the currently in-production recommender system and state-of-the-art travel destination recommender systems and we demonstrate its effectiveness.

2 RELATED WORK

Application of *multi-task learning* (MTL) in recommender systems, knowledge graph-based recommender systems and travel destination recommendations are different research areas related to our work. In this section, we briefly discuss related works in these different areas.

2.1 Multi-task learning for recommendation

Some research work focused on integrating MTL algorithms with traditional CF models such as matrix or tensor factorization [19, 34] in order to generate explainable recommendations. However, these factorization-based models cannot fully exploit the information available in the knowledge graph. In [20], the authors proposed a learning framework composed of two auxiliary tasks (click-through rate and conversion rate optimization) to deal with the extreme data sparsity problem of conversion rate optimization. In [12], the authors proposed a MTL framework to learn simultaneously parameters of two recommendation tasks namely ranking task and rating task. In [2], to deal with the sparsity of the interaction matrix, the authors used MTL to train the model for a combination of content recommendation and item metadata prediction. Similarly to these previous works, we use a neural network with shared parameters learned through different tasks as model architecture. In [33], the authors propose a neural network-based MTL algorithm to predict not only user-item interactions but also missing links in a knowledge graph. Similarly, in [37], the authors mixes a relational modelling algorithm with a recommendation one in a MTL fashion based on a neural network. Nevertheless, the models proposed in the two above-mentioned works do not incorporates literals, thus missing a valuable opportunity for data enrichment. In the opposite, KGMTL4Rec takes into account several types of inputs which constitutes its main strength in comparison with existing MTL algorithms for recommendation.

2.2 Knowledge graph-based recommender systems

In recent years, *knowledge graphs* (KGs) [23] have been used in recommender systems in order to overcome the problem of user-item interactions sparsity and the cold start problem which CF methods suffer from by leveraging properties about items and users and representing them in one single data structure. *Embedding-based* methods which are a subclass of knowledge graph-based recommender systems consist in pre-processing a KG with knowledge graph embedding algorithms [4] and then, incorporating the learned entity embeddings into a recommendation framework [8, 23, 39]. By using knowledge graph embedding techniques, it is now possible to turn virtually any type of information into a vector which the system can learn. One remarkable thing about knowledge graph-based recommender systems is

[1]https://gitlab.eurecom.fr/amadeus/KGMTL4Rec
[2]https://gitlab.eurecom.fr/amadeus/KGMTL4Rec/ontology

their ability to make use of the KG structure to provide better recommendations [31]. However, the above-mentioned methods do not consider multi-typed knowledge graphs that contain multi-valued literals such as categorical values, numerical values, dates, texts, etc. In our work, we propose a model that incorporates heterogeneous information from a multi-typed knowledge graph to recommend travel destinations.

2.3 Travel destination recommendation

Searching for the right destination, or for an hotel or an event can be a complex decision-making process for travelers. For this reason, the recommendation system research community has worked to enhance the search experience for travelers. Early works have focused on personalized techniques in order to provide recommendation based on users' preferences and interests [27]. In [18], the authors proposed PersonalTour a recommender system which is used by travel agencies to find suitable travel packages in accordance with the customer preference. In [29], the authors introduced MyTravelPal, a system providing travel destination recommendations in accordance with the affinity to user areas of interest. In [16], a Naive Bayes model is used to recommend travel destinations in a hotel booking platform based on multi criteria rating data provided by previous users. More recently, in [8], we have shown how to use Semantic Trails Dataset knowledge graph[3] which contains user check-ins in various cities around the world to build location-based embeddings and use them as input of a neural network trained to recommend travel destinations. However, not all the data used as input of the neural network comes from a single data structure. This represents the major difference between our work and that of [8].

2.4 Dataset for tourism recommendation

Several tourism recommendation use cases have been addressed in recent years and consequently a number of datasets have been made public in order to replicate results or even improve on existing findings. In [1], the authors collected a very large-scale hotel recommendation dataset, based on TripAdvisor[4], containing 50 million reviews on hotels. In the hotel booking domain, Trivago[5] has released a public dataset of hotel search sessions as part of the ACM RecSys 2019 Challenge[6], with the goal to build a recommender systen that predicts which hotels (items) the user has clicked on among the search results provided by the metasearch during the last part of the user session. In [21], the authors used Location Based Social Networks to build users' trails, where a trail is a succession of check-ins (the user share his/her location) made by a user in a venue when visiting a city. The Booking.com[7] platform recently released a dataset for the next destination recommendation task as part of the ACM WSDM 2021 WebTour[8] considering different contextual information related to the hotels bookings. The dataset released for this challenge is completely anomymized. Hence, it cannot be used in our work since destinations (referenced by ids) are unknown. To the best of our knowledge, there is no public available dataset that addresses the task of travel destination recommendation that can benefit from the type of data augmentation we are proposing in this work. We describe the experimental dataset we used in this paper later in Section 4.1.

[3]https://figshare.com/articles/dataset/Semantic_Trails_Datasets/7429076
[4]https://www.tripadvisor.com/
[5]https://www.trivago.com/
[6]https://recsys.trivago.cloud/challenge/
[7]https://www.booking.com/
[8]https://www.bookingchallenge.com/

3 PROBLEM FORMULATION

The aim of our work is to build a recommender system that suggests a ranked list of destinations where travelers would like to go to, as shown in Figure 1. More precisely, our objective is to address an inspiration recommendation scenario:

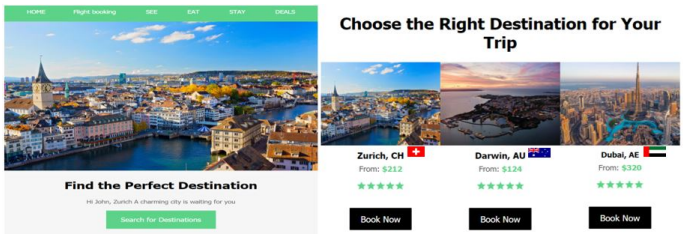

Fig. 1. Top-3 travel destinations recommendation included in a marketing email.

Provide leisure travelers with travel destinations that they have never visited yet. We consider past bookings of travelers, booking contexts and travelers' and destinations' metadata as information to be used in our recommender system. These information are collected and stored in the knowledge graph described in Section 4.1. The task of recommending the next travel destination to a traveler is formulated as a link prediction task in a knowledge graph. In this work, we address the following research questions:

RQ 1: What is the benefit of using a knowledge graph as a common data structure containing all the input information of the recommender system?

RQ 2: Given the heterogeneous nature of the information included in the knowledge graph (numerical values, dates, texts, etc.), what is the best performing approach for travel destination recommendation?

4 KGMTL4REC

In its most simple form, a recommender system is typically built in three consecutive stages [14]: *information collection* which consists in our case in building the knowledge graph; *learning* which corresponds to the multi-task learning algorithm we propose; and *recommendation* which corresponds to the recommendation scoring function presented later in this section. In the following, we detail these three stages for building our recommender system.

4.1 Knowledge Graph Construction

We work on a real-world production dataset of bookings from the T-DNA database[9]. Each booking contains one or several air ticket purchases, and is stored using Passenger Name Record (PNR) information. The PNR is created at reservation time by airline reservation system and contains information about the purchased air ticket (e.g. travel itinerary, payment information), traveler demographics and additional services (e.g. preferred seat, extra bag) if purchased. The dataset we consider contains 486.000 bookings from November 2018 to December 2019, made by 40.965 unique travelers and covering 136 different destinations. Our airline travel KG defines 5 types of entities, namely:

- **Traveler**: A traveler is identified uniquely by a T-DNA id. A traveler has a booking history of purchases (e.g. air tickets). An instance of traveler is a `schema:Person`[10].

[9]T-DNA: Traveler DNA is a private database which contains bookings of travelers over a dozen of airlines. The dataset used in the experiments is GDPR compliant and do not include any personal identifiable information.
[10]The prefix `schema` is used for concepts defined by https://schema.org

- **Trip Reservation**: A trip reservation (PNR) represents the booking of all travelers contained in the PNR. It contains information such as the number of passengers, the destination, etc.
- **Journey**: A journey is linked to a trip reservation. Each journey has a stay duration, a departure and an arrival airport.
- **Air Ticket**: An air ticket is contained in a PNR and contains flight and transactional information.
- **Airport**: It represents the airport where the traveler travels to. An airport serves one or several cities.

We have designed an ontology that we published at http://bit.ly/kg-ontology. A destination where a traveler traveled to is described by a property which we name `travelTo`. The objective of the recommender system is to predict the correct links labeled by the property `travelTo` between travelers and destinations.

In addition to this Airline Travel KG, we make use of the property `owl:sameas` to enrich the knowledge graph with destinations metadata such as the number of inhabitants of a city, the GDP of the country to which the city belongs, etc. In practice, we re-use the Wikidata[11] knowledge graph to capture general information about destinations through wikidata properties (e.g. 'P2046', 'P1081'), the Semantic Trails Dataset (STD) [12] knowledge graph (see Section 2.3) to better understand what characterizes a city the most by using the DBpedia[13] property named 'category' and Wikipedia textual description of the travel destinations to populate our original airline travel KG. In the end, the KG used to tackle our recommendation task contains 48 different properties, ~ 13.7 million edges (~ 634.000 nodes) of which ~ 11.9 Millions come from the Original Airline Travel KG (32 Properties about PNRs, travelers' information, etc.), ~ 1.7 Millions from the STD knowledge graph (5 properties) and ~ 100K from Wikidata (11 properties) and finally ~ 486K edges are travel interactions (property `travelTo`).

In Figure 2, an excerpt of the KG is depicted, where a Singaporean traveler, born on "1994-03-27" booked a one-way flight from Kuala Lumpur to Melbourne (the property `travelTo` coming from the traveler points at Melbourne airport).

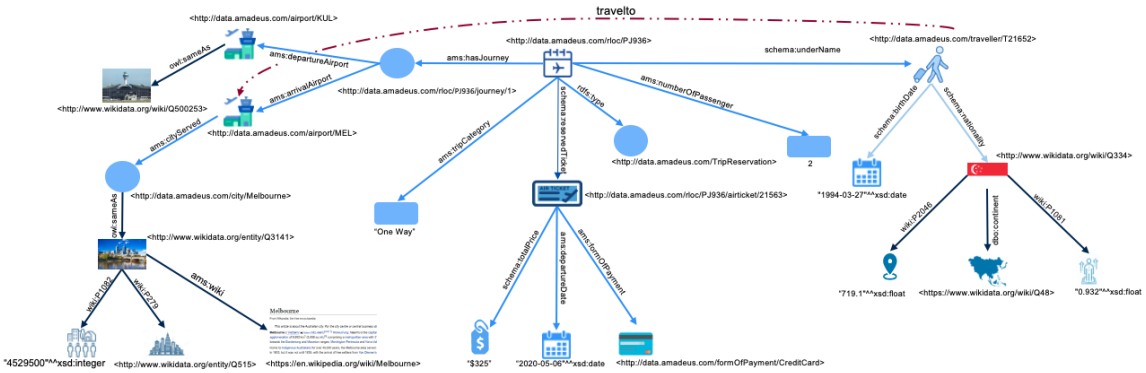

Fig. 2. Excerpt of the knowledge graph representing a traveler included in a Trip reservation through the property `schema:underName`, as well as other properties and relations to other entities. Literals are represented in blue rectangle, whereas other entities are represented in blue circle. In this depiction, some properties which links travelers, trip reservations, air tickets, travel destinations are represented as an example, but more properties are included in the graph.

[11] https://www.wikidata.org/
[12] http://std.eurecom.fr/
[13] https://www.dbpedia.org/

4.2 Multi-task learning algorithm for Multi-typed Knowledge Graph

As mentioned in Section 1, MT-KGNN [32] has recently proven to be an effective approach to deal with non-discrete values in knowledge graphs for representation learning. These authors proposed a multi-objective neural network model trained using a multi-task learning algorithm that includes two regression tasks to predict numerical attributes of KG entities and one classification task to predict when a triplet (head, relation, tail) holds in the KG. In our work, we propose to extend the MT-KGNN model by adding a sub-network called DescNet (Figure 3) that predicts the correct entity described by a textual description given as input of DescNet. Inspired by the DKRL model proposed in [36], we decide to use a convolutional neural network to reduce the dimension of word vectors of the textual descriptions and to train DescNet sub-network along with two other sub-networks (StructNet & AttrNet). We present the model architecture of KGMTL4Rec in Figure 3. As shown in the figure, there are three different learning tasks namely 'Regression' used to capture the information that is contained in entities' attributes, 'Binary Classification' used to capture the structural aspect of the knowledge graph (entities connections), 'Multi-label Classification' used to capture information about entities (more precisely destinations in this work) descriptions.

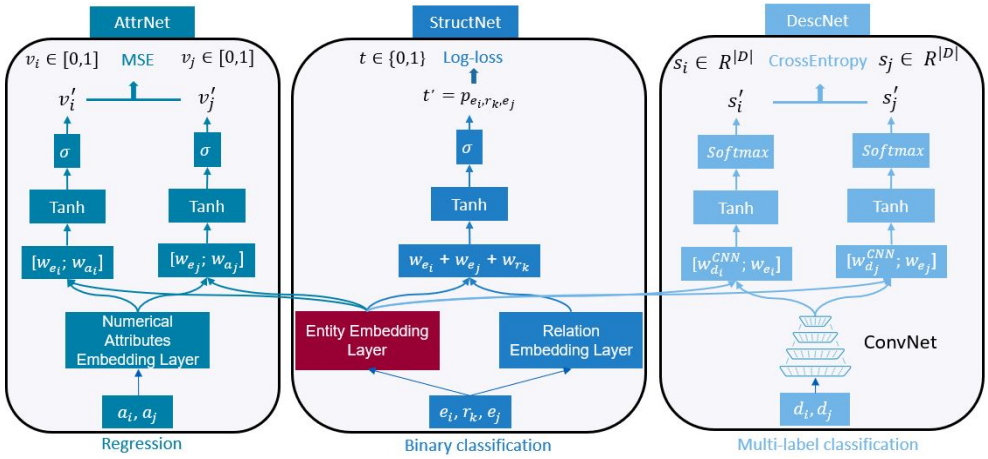

Fig. 3. KGMTL4Rec Architecture: A neural network composed of three sub-networks, each sub-network being specialized in a learning task. The same color is used for different elements of a sub-network (e.g. Turquoise color for AttrNet). Red color is assigned to the 'Entity Embedding Layer' as its weights are shared across the different sub-networks. See section 4.2 for more details

We describe below the different learning tasks and present the multi-task learning algorithm used to train KGMTL4Rec. For the remaining of this section we refer to (e_i, r_k, e_j) as a triplet in the knowledge graph.

Structural Learning (StructNet): The first learning task of KGMTL4Rec corresponds to a binary classification task which is used to predict whether or not the triplet (e_i, r_k, e_j) exists in the knowledge graph. Each element of the input triplet (e_i, r_k, e_j) of StructNet is first passed into an embedding lookup layer. Then, the embeddings $(w_{e_i}, w_{r_k}, w_{e_j}) \in R^d$ are summed and passed into a *hyperbolic tangent* (tanh) nonlinear layer. Finally, a sigmoid linear layer is added to compute the probability $p_{e_i,r_k,e_j} = P((e_i, r_k, e_j) \in T_r)$, where T_r is the set of existing triplets in the knowledge graph. More formally, the probability p_{e_i,r_k,e_j} is computed as follows:

$$p_{e_i,r_k,e_j} = g_{StructNet}(e_i, r_k, e_j) = \sigma(\vec{v}_h^s \ \ tanh(\mathbf{V}_{h,d}^s \ \ (w_{e_i} + w_{r_k} + w_{e_j}) + b_h^s) \tag{1}$$

where $\mathbf{V}^{s}_{h,d} \in R^{h \times d}$ and $\vec{v}^{s}_{h} \in R^{h}$ are parameters of StructNet and b^{s}_{h} is the scalar bias of the hidden layer, h being the size of the hidden layer. We use logistic loss as loss function for this binary classification task. It is important to note that unlike ER-MLP[9], in StructNet, we compute the sum of $w_{e_i}, w_{r_k}, w_{e_j}$ embeddings instead of concatenating them, as it has shown better performance in our experiments.

Numerical Attribute Learning (AttrNet): The second learning task of KGMTL4Rec is a regression task, where the objective is to predict the correct numerical value of an entity attribute (e.g. the price of an air ticket). AttrNet takes as input the attributes a_i and a_j linked to e_i and e_j entities. The embedding $w_{a_i} \in R^{m}$ is concatenated with w_{e_i} and $w_{a_j} \in R^{m}$ with w_{e_j}. Then, the concatenated vectors are passed into a tanh nonlinear hidden layer and finally passed into a sigmoid linear layer to compute the estimated numerical values $v^{'}_{i}$ and $v^{'}_{j}$. More formally, the estimated value $v^{'}_{i}$ is computed as follows:

$$v^{'}_{i} = g_{AttrNet}(e_i, a_i) = \sigma(\vec{v}^{a}_{h} \ \ tanh(\mathbf{V}^{a}_{h,md} \ \ [w_{e_i}; w_{a_i}] + b^{a}_{h}) \tag{2}$$

where $\mathbf{V}^{a}_{h,md} \in R^{h \times (m+d)}$ and $\vec{v}^{a}_{h} \in R^{h}$ are parameters of AttrNet and b^{a}_{h} is the scalar bias of the hidden layer.

The Mean Squared Error (MSE) is used as a loss function for AttrNet. Unlike what was done in MT-KGNN [32], we use only one single AttrNet regardless if an attribute is linked to the tail or the head entity of a triplet.

Text description Learning (DescNet): The third learning task of KGMTL4Rec is a multi-class classification task, where the objective is to predict the correct entities e_i and e_j given the input text descriptions of destinations d_i and d_j. The first part of DescNet is a convolutional neural network (CNN) composed of one convolutional layer and a max-pooling layer used to reduce the dimension of input word vectors. Similarly to what is done in [8], we assign to each word of the text description d_i and d_j a weighted tf-idf pre-trained word vector from fasttext [11]. The CNN is then fed with $w_{d_i} \in R^{|d_i| \times k}$ and $w_{d_j} \in R^{|d_j| \times k}$, vector representations of d_i and d_j, where $|d_i|$ and $|d_j|$ represent the length of the text descriptions d_i and d_j and k the dimension of word vectors. Finally, the output vectors of the CNN $(w^{CNN}_{d_i}, w^{CNN}_{d_j})$ are passed into a tanh nonlinear hidden layer, and then passed into a Softmax linear layer to compute the estimated vectors $s^{'}_{i}$ and $s^{'}_{j} \in R^{|D|}$, D being the set of travel destinations. More formally, $s^{'}_{i}$ is computed as follows:

$$s^{'}_{i} = g_{DescNet}(d_i) = Softmax(\vec{v}^{d}_{h} \ \ tanh(\mathbf{V}^{d}_{h,k} \ \ w^{CNN}_{d_i}) + b^{d}_{h}) \tag{3}$$

where $\mathbf{V}^{d}_{h,k} \in R^{h \times k}$ and $\vec{v}^{d}_{h} \in R^{h}$ are parameters of DescNet and b^{d}_{h} is the scalar bias of the hidden layer.

Note that the learning task is performed twice for the head and the tail entity of the input triplet (e_i, r_k, e_j) for each of the learning tasks in AttrNet and DescNet. The reason behind this architectural choice lies in the fact that **each** entity in the triplet can simultaneously have a description and an attribute.

Multi-task learning algorithm: We adopt an alternating learning strategy for the five learning tasks. More formally, for each epoch, we run the following:

- Sample mini-batch of positive and negative triplets (e_i, r_k, e_j) from the knowledge graph, train StructNet and update KGMTL4Rec parameters by back-propagation according to Eq 1.
- Sample mini-batch of numerical attributes a_i and a_j and their corresponding numerical values v_i and v_j, train AttrNet and update KGMTL4Rec parameters by back-propagation according to Eq 2.
- Sample mini-batch of textual descriptions d_i and d_j of e_i and e_j entities, train DescNet and update KGMTL4Rec parameters by back-propagation according to Eq 3.

In the experiments, we compare the alternating learning strategy with the weighting loss strategy [5, 38] where the different losses of the sub-networks are summed so that the sum of the losses is back-propagated through KGMTL4Rec.

4.3 Recommendation scoring function

As mentioned in Section 3, the task of recommending destinations to travelers is formulated as a link prediction task in the knowledge graph. Therefore, in order to compute the probability of recommending a destination e_d to a traveler e_t, we use StructNet sub-network and compute the score of the triplet (e_t, 'travelTo', e_d) comprising the traveler e_t, the destination e_d, and the property 'travelTo'. The recommendation scoring function is defined as follows:

$$f_{recommendation}(e_t, e_d) = g_{StructNet}(e_t, travelTo, e_d) \tag{4}$$

5 EXPERIMENTAL EVALUATION

In this section, we present the dataset used to conduct our experiments. Then, we present some baseline models we have implemented to compare our model with and the settings of the experiments. Finally, we discuss the results obtained in the experiments.

5.1 Dataset

For the experiments, we use the private dataset described in Section 4.1. It is important to note that due to the specificity of our recommendation task 'recommending **new** travel destinations for **leisure** purpose', the amount of data used in the experiments is significantly reduced. Indeed, The original dataset used to build the knowledge graph comes from a major partner airline and counts more than 10 million bookings in one calendar year. In this work, we focus only on leisure trips, which corresponds to approximately 56% of the bookings similarly to what has been done in [8]. Furthermore, the dataset that is used to train the recommender system is reduced as we only consider travellers who have made at least two bookings (for evaluation purposes), resulting in 486.807 travel interactions. The characteristics of the dataset are summarized in Table 1.

Table 1. Statistics of the experimental dataset.

#travels	#travelers	#destinations	Sparsity ρ
486 807	40 965	136	91.26%

In Figure 4, we plot an histogram that represents the number of visits per travel destination as a percentage of the total number of visits (#travels) for the top-10 most visited destinations. This histogram shows the high popularity of certain travel destinations which is accounted for in the experiments by comparing the performance of our model with the system currently in production which some airline partners use and that recommends this top-10 list of popular destinations regardless of the traveler. In Figure 5, we plot an histogram representing the number of travelers (as a percentage of total number of travelers) per historical travels. In the experiments, we compare the performance of our model with respect to the number of historical travels per traveler.

We use the KG resulting from the dataset mentioned above and described in Section 4.1 for KGMTL4Rec and knowledge graph-based baseline models. In Table 2, we present some characteristics about the KG used in the experiments.

Table 2. Statistics of the experimental knowledge graph.

#Nodes	#Edges	#Properties	#Trip Reservations
634 254	~ 13.7 M	48	423 427

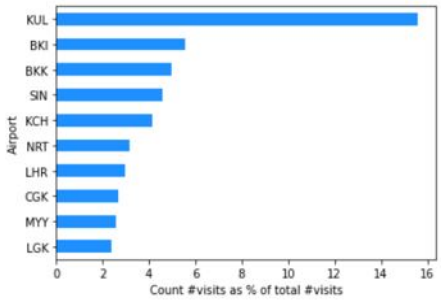

Fig. 4. Top-10 Most visited travel destinations (airports). Each airport is named according to its IATA code.

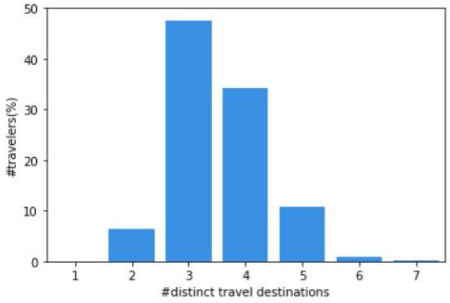

Fig. 5. Histogram showing the number of travelers per number of distinct historical travel destinations.

5.2 Evaluation protocol and metrics

Widely used in the literature [8, 13, 26], the *leave-last-out* protocol suggests to select the latest interaction as the test set and use the remaining data in the training/validation set. We use this protocol to evaluate the performance of KGMTL4Rec and also to compare it with the different baseline models. Our dataset is temporally sorted so that the latest travel corresponds to the most recent destination visited by a traveler, in order to represent the notion of recommending the 'next' travel destination during evaluation. For each traveler, we rank all destinations except the ones that are already visited by the traveler and truncate the list at 10, as 10 destinations are included in the email sent to the travelers. To validate our model, we apply a cross-fold validation to the training dataset (k=5, a split of 80% for training and 20% for validation). The split between training and validation set is performed randomly on travels in order to avoid a seasonality effect which is usually occurring in the travel industry.

The output of the recommender system is a ranked list of 10 destinations, where at best, one element of the 10 recommended destinations is a relevant one and corresponds to the 'next' travel destination of the traveler. Given that, we think it is judicious to use the Hit Rate metric to measure whether or not the relevant destination is in the top-10 list and we use Mean Reciprocal Rank metric to capture how well the hit is ranked in the list. The two metrics are defined as follows:

- **HR@K**:

$$HR@K = \frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{K} hit(t, d_j) \tag{5}$$

- **MRR@K**:

$$MRR@K = \frac{1}{n}\sum_{t=1}^{n}\sum_{j=1}^{K} \frac{1}{rank(rel_t)} \tag{6}$$

where n represents the number of travelers, K the length of the ranked list and $hit(t, d_j)$ is equal to 1 if the traveler t traveled to the destination d_j. In equation 6, $rank(rel_t)$ is the rank of the relevant destination where the traveler t has traveled to. The rank is only considered if the relevant destination is in the top-K list.

5.3 Baseline models and parameter settings

We implement a wide list of baseline models to compare with our KGMTL4Rec model. More specifically, the baseline models include collaborative filtering, *context-aware*, *hybrid* and knowledge graph-based recommender systems.

Following the experimental work conducted in [8], this represents the state-of-the-art recommender systems for travel destination recommendation. We describe the main baseline models we used and we list the different types of information used in each algorithm:

- **BPRMF [26]:** *BPRMF* is a *Matrix Factorization* method tailored for implicit feedback where the authors propose to minimize a pairwise ranking loss rather than minimizing a mean squared error between the predicted and the observed 'rating' as usually done in Matrix Factorization algorithm. *BPRMF* uses only user-item interactions as input.
- **NCF [13]:** *Neural Collaborative Filtering* is a state-of-the-art CF method. It combines the (user, item) interaction as input of a multi-layer perceptron and a single layer perceptron that models the matrix factorization method. *NCF* uses only user-item interactions as input.
- **FM [25]:** *Factorization Machines* was proposed to incorporate contextual information in the recommender system. The author propose a method that computes not only users' and items' latent vectors but also contextual features latent vectors. *FM* uses contextual information in addition to user-item interactions as input.
- **WDL [6]:** *Wide & Deep Learning* model is a hybrid recommender system. It is a deep learning based recommender system that combines a deep component (feed forward neural network) plus a wide component that can be seen as a linear model that computes cross products between input features. *WDL* uses contextual and content information in addition to user-item interactions as input.
- **DKFM [8]:** *Deep Knowledge Factorization Machines* combines Factorization Machines in order to represent contextual information and WDL that takes as input user-item interactions and metadata information about the items and users. *DKFM* uses all type of information available in the knowledge graph.
- **NTN [30]:** *Neural Tensor Network* is a neural network based method for representation learning in knowledge graphs [35]. Given a fact (h, r, t), it first projects entities to their vector embeddings in the input layer and then predicts the existence of this fact in the knowledge graph. Similarly to StructNet (see section 4.3), we rank destinations based on NTN output score. *NTN* uses all type of information available in the knowledge graph.
- **TransE [4]:** *TransE* is the most used translational distance model [35]. Given a fact (h, r, t), the relation is interpreted as a translation vector r so that the embedded entities h and t can be connected by r with low error, i.e., $h + r \approx t$ when (h, r, t) holds. Similarly to [24], we use TransE scoring function $f_r(h, t) = -||h + r - t||$ to produce the ranked list of destinations. *TransE* uses all type of information available in the knowledge graph.
- **CKE [39]** *Collaborative Knowledge base Embedding* is a two stages approach that consists in first computing the embeddings coming from a knowledge base composed of structural knowledge, image and text representing the items, then use the generated embeddings as input of a CF algorithm. In this paper, we implement the structural and textual modules in addition to the CF algorithm. *CKE* uses all type of information available in the knowledge graph.

We implement our model KGMTL4Rec using Pytorch[14] as it provides more easiness for the implementation of new neural network architectures and use Pykg2vec[15] library for knowledge graph-based models. Finally, we use Tensorflow[16] to implement the neural network baseline models. We use Xavier uniform initializer to randomly initialize the models parameters and we use a mini-batch optimization technique based on Adam [15] optimizer to train all the models. To tune the hyper-parameters of our model and the baseline models, we use the validation set mentioned

[14] https://pytorch.org/
[15] https://pykg2vec.readthedocs.io/
[16] https://www.tensorflow.org/

above (Section 5.2). We apply grid-search algorithm on the implemented models using the following values: the entity embedding size $d \in \{16,32,64,128,256\}$, the batch size $\in \{128,256,512,1024\}$, the number of epochs $\in \{10,20,50,100,200\}$, the learning rate $\lambda \in \{0.00001,0.0001,0.0003,0.001,0.003,0.01,0.1\}$ and negative samples Ns $\in [2,10]$.

5.4 Results

In Table 3, we present the recommendation performance of KGMTL4Rec and the baseline models with respect to HR@10 and MRR@10. The results reported in Table 3 correspond to the performance of the different models based on the best performing hyper-parameters. We report the mean and standard deviation of HR@10 and MRR@10 over 5 different seeds due to the random initialization of neural networks parameters.

Table 3. Experimental results.

(a) Recommendation performance of CF, hybrid and context-aware recommender systems.

Model	HR@10	MRR@10
Item-pop	0.5168	0.2634
Item-KNN [28]	0.3223	0.1367
BPRMF [26]	0.5698 ± 0.002	0.3036 ± 0.0004
NCF [13]	0.5132 ± 0.008	0.2994 ± 0.0010
FM [25]	0.5986 ± 0.003	0.3401 ± 0.0001
WDL [6]	0.6301 ± 0.005	0.3472 ± 0.0003
DKFM [8]	**0.6619** ± 0.007	**0.3901** ± 0.0006

(b) Recommendation performance of knowledge graph-based recommender systems.

Model	HR@10	MRR@10
NTN [30]	0.3096 ± 0.002	0.1511 ± 0.001
SME [3]	0.3746 ± 0.001	0.1992 ± 0.0004
TransE [4]	0.4548 ± 0.0005	0.2268 ± 0.0001
TransR [17]	0.4031 ± 0.0009	0.1883 ± 0.0001
ER-MLP [9]	0.6218 ± 0.002	0.3559 ± 0.0028
CKE [39]	0.6493 ± 0.003	0.3865 ± 0.001
KGMTL4Rec	**0.7109** ± 0.013	**0.4254** ± 0.0083

It is important to note that not all recommender systems use the same input information. In fact, recommender systems which use not only traveler history but also other types of information as input such as DKFM or WDL tend to perform better than simple Collaborative Filtering models such as ImplicitMF, NCF or Item-KNN as shown in sub-table (a). Similarly to DKFM, knowledge graph-based recommender systems represented in sub-table (b) make use of all the information mentioned in Section 3. It is therefore legitimate to compare KGMTL4Rec with DKFM, where we clearly observe that KGMTL4Rec performs better with respect to HR@10 and MRR@10. KGMTL4Rec is not only outperforming DKFM model but also the other knowledge graph-based recommender systems represented in sub-table (b). The major difference between KGMTL4Rec and the other knowledge graph-based recommender systems, is that KGMTL4Rec uses each type of information optimally in one of the sub-networks defined in Section 4.2, while models like TransE, NTN or even CKE (that uses TransE to generate structural embeddings) consider numerical values as a separate entity, which not only increases considerably the cardinality of entities set considered in this type of method, but also considers equal numerical values as the same entity: it is not correct to consider 12 'years old' and 12 'days' as the same entity.

In what follows, we take an excerpt (~20%) from the original knowledge graph described in Table 2 in order to conduct additional experiments thanks to the reduced size of the dataset. All the results that follow are based on this excerpt (Table 4).

In Table 5, we report the performance of our model compared to the best performing models when we use different types of input information, so the knowledge graph is reduced to keep only the information needed in each experiment to be fairly comparable to other models:

Table 4. Statistics of the sample knowledge graph.

#Nodes	#Edges	#Properties	#Trip Reservations
125 610	~ 2.7 M	48	35698

Table 5. Performance of KGMTL4Rec compared to best performing models on specific type of input data. *: All Information mentioned in section 3.

Input Data	Collaborative Information		Content & Collaborative Information		All Information*	
Model	BPRMF	**KGMTL4Rec**	WDL	**KGMTL4Rec**	DKFM	**KGMTL4Rec**
HR@10	0.5462	**0.5623**	0.6001	**0.6508**	0.6464	**0.6907**
MRR@10	0.3020	**0.3153**	0.3472	**0.4061**	0.3856	**0.4189**

Figure 6 shows the performance of the models represented in Table 5 with respect to the number of iterations used to train the models. We use the same learning rate for all the models ($lr = 0.00003$) presented in Figure 6. We observe that the most effective updates are occurred in the first 3 iterations for all the models except for DKFM where the convergence requires more iterations. In addition, we notice that there is a significant difference of HR@10 and MRR@10 in the first iteration (iteration 0) for the different models. Moreover, it is important to note that for KGMTL4Rec, we do not get the best value of MRR@10 and HR@10 in the same iteration.

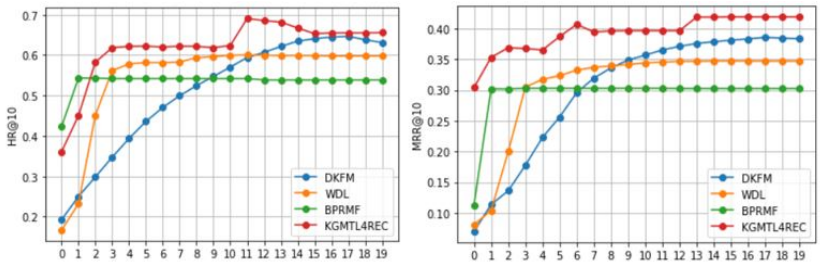

Fig. 6. Performance of the 4 main models (presented in table 5) with respect to the number of iterations.

6 DISCUSSION

We first start by performing an ablation study which consists in removing some input information from the knowledge graph and using only some sub-networks of KGMTL4Rec. Then, we study the influence of the travel history of travelers (number of historical travels) on the performance of KGMTL4Rec. We observe the convergence time of KGMTL4Rec with respect to two different MTL strategies. Finally, we perform a qualitative analysis of KGMTL4Rec recommendations and investigate the impact of KGMTL4Rec hyper-parameters on the performance of the model.

Ablation Study: Table 6 shows the performance of KGMTL4Rec with respect to the information included in the knowledge graph. In the first row of the table, we present the results of KGMTL4Rec when we consider neither the STD knowledge graph nor the textual information from Wikipedia, nor the numerical literals included in the Airline Travel KG (e.g., the number of passengers in a reservation, the ticket price, etc.), and in this case, we use only the sub-network StructNet to train the model. Then, for each of the rows that follow, we add incrementally one of the preceding removed information. We observe that the results are the best when we use all possible information in the KG, and notice that the large gap between the results is reduced when we consider the use of numerical values.

Table 6. Performance of KGMTL4Rec model based on the information contained in the knowledge graph.

Numerical literals	STD KG	Wikipedia	Sub-networks	HR@10	MRR@10
No	No	No	StructNet	0.5884	0.3264
Yes	No	No	StructNet, AttrNet	0.6508	0.4061
Yes	Yes	No	StructNet, AttrNet	0.6781	0.4119
Yes	Yes	Yes	StructNet, AttrNet, DescNet	**0.6907**	**0.4189**

Influence of travel history: Collaborative Filtering algorithms which rely only on users' past interactions perform naturally better when we have more history about the users. We study the performance of KGMTL4Rec model and DKFM model presented in Table 5 with respect to the number of historical travels per traveler. More formally, we compute HR@10 and MRR@10 for travelers which traveled in N_{hist} different destinations in their past ($N_{hist} \in [1, 5]$). We observe in Figure 7 more variation of HR@10 and MRR@10, when we vary the number of historical travels for DKFM than for KGMTL4Rec. Indeed, the standard deviation of the different values of HR@10 for DKFM is equal to 2×10^{-2}, while for KGMTL4Rec it is equal to 5×10^{-3}. For MRR@10, the standard deviation is equal to 2.5×10^{-2} for DKFM, while for KGMTL4Rec it is equal to 6×10^{-3}. These results demonstrate that our model KGMTL4Rec is more resilient to variation of the traveler history than DKFM.

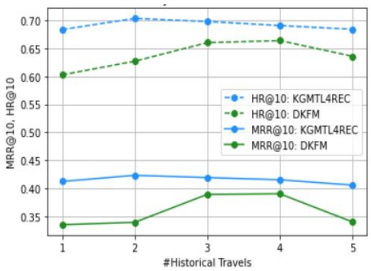

Fig. 7. Performance of KGMTL4Rec with respect to the number of Historical travels per traveler.

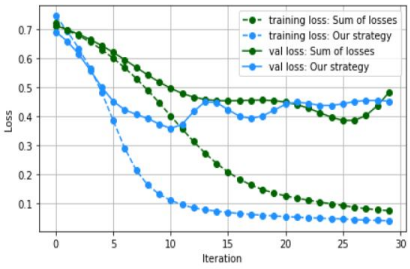

Fig. 8. Training and Validation loss with respect to the number of iterations for two different MTL strategies.

Multi-task learning strategy: While in most multi task learning algorithms the back-propagation is performed based on the sum of the losses of the different tasks [5, 38], we decide to use another strategy which is to perform a back-propagation to update the weights of our model for each learning task, as we do not think judicious to share the same loss across the different sub-networks of KGMTL4Rec when updating the model parameters. Indeed, when training a multi-task learning algorithm, it is possible that too much loss influences the overall loss if the sum of the losses strategy is used to back-propagate on the neural network, which can lead to a too slow convergence of the algorithm. We demonstrate in Figure 8 that our learning strategy converges faster than the strategy used in [5, 38]. Indeed, we observe that the 'sum of losses' strategy needs 12 more iterations than our strategy for the training loss to be equal to 0.1. Moreover, for the validation loss, our strategy needs 10 iterations to converge to a value of 0.38 while for the 'sum of losses' strategy 13 more iterations are needed.

Hyper-parameters Sensitivity: We investigate the influence of some hyper-parameters on the performance of KGMTL4Rec. In Figure 9, we report the score of HR@10 and MRR@10 when we vary the learning rate λ and the entity embedding size d as specified in Section 5.3. We observe in Figure 9 that when increasing d, the performance is initially

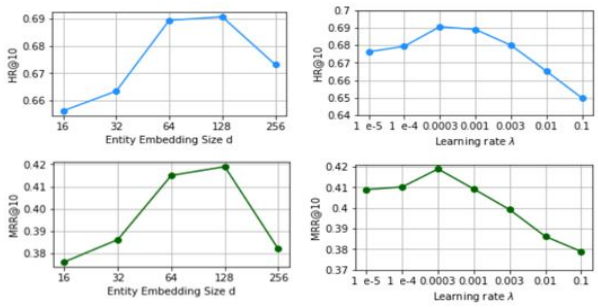

Fig. 9. Performance of KGMTL4Rec with respect to the entity embedding size and the learning rate λ.

improved because embeddings with larger size can encode more useful information, but drops after d = 128 due to possible overfitting. The same pattern is observed when varying λ, indeed the HR@10 and MRR@10 scores increase until $\lambda = 0.0003$ as the use of a higher λ does not allow to find the optimal loss.

Ethical Considerations: The database used in this work is GDPR[17] compliant and the dataset used for the experiments does not contain any personal identifiable information. It is important to note that from a business perspective, the objective of this work is not only to improve the accuracy of recommended destinations but also to ease travelers' search experience by sending them personalized travel destinations through email marketing campaigns in order to avoid spamming them. Bias is not an aspect which is studied in-depth in this work. However, by computing the relatedness score between top-10 recommended destinations and two entities that imply bias namely gender and nationality, we observe minor discrepancies: In average a difference of $5.3e-2$ for nationality entity and $3.6e-2$ for gender entity.

7 CONCLUSION

In this work, we propose KGMTL4Rec, a multi-task learning model designed to consider not only knowledge graph entities but also numerical and text literals in order to recommend personalized travel destinations to airlines' customers through email marketing campaigns. Our model is based on a neural network architecture which can incorporate different types of information available in the knowledge graph. We conduct several experiments to address the research questions mentioned in Section 3. Our model is capable of predicting the missing links 'travelTo' in the knowledge graph with a HR@10 of ~ 0.69 (*RQ2*). Additionally, we demonstrate, through an in-depth comparison between KGMTL4Rec and DKFM, the valuable contribution of using the knowledge graph as a common data structure to represent the heterogeneous information used for travel destination recommendation (*RQ1*). In future work, the group plans to test the model in production and evaluate user engagement through online metrics such as click-through rate or conversion rate. More formally, the challenge is to validate whether the most accurate offline model is the most engaging online and, from a business perspective, to assess whether the model will boost travel bookings.

REFERENCES

[1] Diego Antognini and Boi Faltings. 2020. HotelRec: a Novel Very Large-Scale Hotel Recommendation Dataset. In *Proceedings of The 12th Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 4917–4923. https://www.aclweb.org/anthology/2020.lrec-1.605

[17] https://gdpr.eu/

[2] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-Task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/2959100.2959180

[3] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94, 2 (01 Feb 2014), 233–259. https://doi.org/10.1007/s10994-013-5363-6

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc., Lake Tahoe, Nevada, United States. https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholm, Sweden, 794–803. http://proceedings.mlr.press/v80/chen18a.html

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[7] Amine Dadoun, Michael Defoin-Platel, Thomas Fiig, Corinne Landra, and Raphaël Troncy. 2021. How recommender systems can transform airline offer construction and retailing. *Journal of Revenue and Pricing Management* (2021). https://doi.org/10.1057/s41272-021-00313-2

[8] Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. 2019. Location Embeddings for Next Trip Recommendation. In *Companion Proceedings of The 2019 World Wide Web Conference* (San Francisco, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 896–903. https://doi.org/10.1145/3308560.3316535

[9] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 601–610. https://doi.org/10.1145/2623330.2623623

[10] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. 2020. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? *Semantic Web* Preprint (2020), 1–31. https://doi.org/10.3233/SW-200404

[11] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Miyazaki, Japan. https://www.aclweb.org/anthology/L18-1550

[12] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and Rate: Multi-Task Learning for Recommender Systems. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 451–454. https://doi.org/10.1145/3240323.3240406

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[14] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* 16, 3 (2015), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[16] Julia Kiseleva, Melanie J.I. Mueller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, Jaap Kamps, Alexander Tuzhilin, and Djoerd Hiemstra. 2015. Where to Go on Your Next Trip? Optimizing Travel Destinations Based on User Preferences. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 1097–1100. https://doi.org/10.1145/2766462.2776777

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, Austin, Texas, 2181–2187.

[18] Fabiana Lorenzi, Stanley Loh, and Mara Abel. 2011. PersonalTour: A Recommender System for Travel Packages. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02 (WI-IAT '11)*. IEEE Computer Society, USA, 333–336. https://doi.org/10.1109/WI-IAT.2011.69

[19] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like It: Multi-Task Learning for Recommendation and Explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 4–12. https://doi.org/10.1145/3240323.3240365

[20] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1137–1140. https://doi.org/10.1145/3209978.3210104

[21] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, and Maurizio Morisio. 2018. Semantic Trails of City Explorations: How Do We Live a City. arXiv 1812.04367. http://arxiv.org/abs/1812.04367

[22] Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. 2020. entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Syst. Appl.* 151 (2020), 113235. https://doi.org/10.1016/j.eswa.2020.113235

[23] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. Entity2Rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *Eleventh ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 32–36.

[24] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. 2018. Translational Models for Item Recommendation. In *The Semantic Web: ESWC 2018 Satellite Events*. Springer International Publishing, Cham, 478–490.

[25] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, 452–461.

[27] Francesco Ricci. 2002. Travel recommender systems. *IEEE Intelligent Systems* (2002).

[28] Badrul Sarwar, George Karypis, Joseph A Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web, WWW 2001 (WWW '01)*. Association for Computing Machinery, Inc, Hong Kong, 285–295. https://doi.org/10.1145/371920.372071

[29] Koceski Saso and Petrevska Biljana. 2013. Empirical Evidence of Contribution to E-Tourism by Application of Personalized Tourism Recommendation System. *Scientific Annals of Economics and Business* 59, 1 (July 2013), 363–374. https://ideas.repec.org/a/vrs/aicuec/v59y2012i1p363-374n25.html

[30] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc., Lake Tahoe, Nevada, United States. https://proceedings.neurips.cc/paper/2013/file/b337e84de8752b27eda3a12363109e80-Paper.pdf

[31] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent Knowledge Graph Embedding for Effective Recommendation. In *12th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 297–305.

[32] Yi Tay, Luu Anh Tuan, Minh C. Phan, and Siu Cheung Hui. 2017. Multi-Task Neural Network for Non-Discrete Attribute Prediction in Knowledge Graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17)*. Association for Computing Machinery, New York, NY, USA, 1029–1038. https://doi.org/10.1145/3132847.3132937

[33] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2000–2010. https://doi.org/10.1145/3308558.3313411

[34] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/3209978.3210010

[35] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (Dec 2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[36] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, Phoenix, Arizona, 2659–2665.

[37] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon Jose. 2019. Relational Collaborative Filtering: Modeling Multiple Item Relations for Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/3331184.3331188

[38] Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2644–2655. https://doi.org/10.18653/v1/N19-1271

[39] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 353–362. https://doi.org/10.1145/2939672.2939673