Article

THOR: A Hybrid Recommender System for the Personalized Travel Experience

Alireza Javadian Sabet [1,2,*,†], Mahsa Shekari [3,†], Chaofeng Guan [2], Matteo Rossi [3], Fabio Schreiber [2] and Letizia Tanca [2]

[1] Department of Informatics and Networked Systems, University of Pittsburgh, Pittsburgh, PA 15260, USA
[2] Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Via Giuseppe Ponzio, 34/5, I-20133 Milano, Italy
[3] Politecnico di Milano, Dipartimento di Meccanica, Via La Masa, 1, I-20156 Milano, Italy
* Correspondence: alj112@pitt.edu
† These authors contributed equally to this work.

Abstract: One of the travelers' main challenges is that they have to spend a great effort to find and choose the most desired travel offer(s) among a vast list of non-categorized and non-personalized items. Recommendation systems provide an effective way to solve the problem of information overload. In this work, we design and implement "The Hybrid Offer Ranker" (THOR), a hybrid, personalized recommender system for the transportation domain. THOR assigns every traveler a unique contextual preference model built using solely their personal data, which makes the model sensitive to the user's choices. This model is used to rank travel offers presented to each user according to their personal preferences. We reduce the recommendation problem to one of binary classification that predicts the probability with which the traveler will buy each available travel offer. Travel offers are ranked according to the computed probabilities, hence to the user's personal preference model. Moreover, to tackle the cold start problem for new users, we apply clustering algorithms to identify groups of travelers with similar profiles and build a preference model for each group. To test the system's performance, we generate a dataset according to some carefully designed rules. The results of the experiments show that the THOR tool is capable of learning the contextual preferences of each traveler and ranks offers starting from those that have the higher probability of being selected.

Keywords: Context-Aware Recommender System; personalization; preferences; user modeling; journey planning; mobility; cold start; classification; clustering

check for updates

Citation: Javadian Sabet, A.; Shekari, M.; Guan, C.; Rossi, M.; Schreiber, F.; Tanca, L. THOR: A Hybrid Recommender System for the Personalized Travel Experience. *Big Data Cogn. Comput.* **2022**, *6*, 131. https://doi.org/10.3390/bdcc6040131

Academic Editors: Konstantinos Kotis and Dimitris Spiliotopoulos

Received: 21 September 2022
Accepted: 1 November 2022
Published: 4 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The recent blazing-fast advancements in big data analysis have unlocked the potential of many novel applications for smart living [1]. In particular, big data techniques can be used to greatly enhance how users experience personalized mobility, for which demand is growing fast [2]. The Internet has become the main channel for travelers to obtain online information before traveling, but they still spend a great effort to find and choose the most desired travel offer(s) among a vast list of non-categorized and non-personalized ones [3].

A Recommender System (RS), a.k.a., Recommendation System, aims at predicting the "preference" of a user about an item [4] and may provide a great help when users have search or selection problems. Recently, there have been various advancements in the field of Context-Aware Recommender Systems (CARS) [5–7]. Indeed, one should also take into account that travel preferences may be significantly influenced by *the context in which a traveler interacts with the system* [8,9]. According to Dey [10], "Context is any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object" and "A system is context-aware if it uses context to provide relevant information and services to the user, where relevancy depends on the user's task". For more information about context-aware systems, we refer the reader to surveys provided in [11–13].

Despite various works that tackled the problem of recommending touristic destinations to travelers [14–16], to the best of our knowledge, no works exist that designed CARS for ranking a list of complete travel offers for travelers.

In this work, we assume that the user has access to an application to ask for travel offers, such as, for example, the Travel Companion (TC) provided by the Shift2Rail ecosystem (see Section 2.1). To provide a personalized experience to travelers, Javadian et al. designed a high-level system architecture in [9] and a contextual preference model in [8] using the Context Dimension Tree methodology [17]. This work aims to implement the recommender core of the TC (or of a TC-like application) based on the contextual preference model presented in [9]. More precisely, this work aims at answering the following research questions (RQs):

RQ1: Using the traveler's historical records, how can we design a personalized preference model which ranks the available travel offers according to the contextual preferences of the traveler?

RQ2: Given a new traveler without any historical records, how can we design an initial personal preference model for them using other travelers' historical data with the most similar characteristics?

To tackle these research questions and provide travelers with a Context-Aware Recommender System, the *contribution* of this work is the development of "The Hybrid Offer Ranker" (THOR), which could be integrated in applications such as the Shift2Rail TC. Unlike the proposed RS in [18], which requires a user-specified list of preferences for filtering and ranking the travel offers, THOR relies only on the historical purchase records of the users while incorporating the user-specified preferences whenever available. THOR assembles various algorithms to create a pipeline that considers the problem of ranking travel offers shown to users according to their preferences as a *binary classification problem* that predicts if the traveler will buy any of the available travel offers or not. First, each traveler is assigned a unique *contextual preference model* built using their data. For this purpose, THOR uses various well-known classification algorithms—i.e., *K-Nearest Neighbors (KNN)* [19], *Support Vector Classifier (SVC)* [20], *Decision Tree (DT)* [21], *Random Forest (RF)* [22], and *Logistic Regression (LR)* [23]—and finds the best set of hyper-parameters. Then, when the system receives a set of travel offers to be shown to the user, it exploits the contextual preference model to determine, for each offer, the probability that the user will buy it and ranks the offers accordingly. Moreover, THOR incorporates the *K-means* [24] and *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* [25] clustering algorithms to identify users with similar profiles and build a preference model for each group of similar users. In the case of a new user without any record, we identify the group that contains the most similar users and use its associated preference model to provide recommendations for the new user. Notice that the clustering and classification algorithms are used independently of one another. More precisely, during each training phase, all training algorithms are executed separately, and the one that produces the model with the best performance—i.e., the highest accuracy—is saved for future use. To test the system's performance (in terms of both computation time and accuracy), we generate an extensive dataset according to some carefully designed rules. The results show that the proposed framework is promising and can provide benefits to existing systems.

The rest of the paper is organized as follows. Section 2 presents state-of-the-art methodologies concerning the problem of designing RSs in the transportation domain. Section 3 details THOR's implementation. Section 4 shows the validation of THOR using different approaches. Finally, Section 5 concludes and presents some future work.

2. Background and Related Work

This section discusses the relevant related work and how this inspired the methodology we propose. To the best of our knowledge, most previous research on RS for the transportation domain focuses on designing a system that aims at recommending the most suitable destination to a traveler; instead, our approach aims to design an RS that, given a destination and a mobility request (i.e., the action in which a traveler uses an application to request possible travel solutions to go from a source to a destination), ranks a set of travel offers satisfying that request according to the user preferences. In Section 2.1 we briefly discuss the ecosystem in which we chose to integrate our proposed system. Section 2.2 discusses the state-of-the-art about RSs in the transportation domain; Section 2.3 investigates various important features (e.g., time, location, travel purpose) contributing to the travelers' preferences employed by state-of-the-art approaches; finally, Section 2.4 investigates techniques to tackle the crucial *cold start* problem.

2.1. Ecosystem

This work is performed within the Ride2Rail (R2R) project [26] (ride2rail.eu, accessed on 10 October 2022), in the frame of Innovation Programme 4 (IP4) of the Shift2Rail (S2R) initiative (www.shift2rail.org, accessed on 10 October 2022)) which aims at enhancing the European Transportation domain. The S2R initiative aims at implementing a collaborative ecosystem through its Interoperability Framework [27] that provides various modules and services such as automated mapping [28,29], data conversion [30], and ontology management [31]. The ecosystem facilitates the interoperability between IP4 services (e.g., Booking, Journey Planning) and Travel Service Providers (TSP) and permits them to interact, share data, and create more complex services while addressing their security and privacy concerns [32]. The ecosystem includes a component, the so-called Travel Companion (TC), which is an application that can run on various devices—e.g., smartphones—and assists travelers before, during, and after each journey.

As mentioned in Section 1, the existence of an application assisting users in setting up their travels is a prerequisite of our approach. The proposed system (THOR) is general and it could be part of any such application. However, the specific application into which the system is integrated has an impact on THOR's implementation. In particular, the set of travel features handled by the application determines the data fed to THOR for learning preference models (see Table 1) and affects the resulting recommendations. THOR was developed to be part of the S2R ecosystem, hence its implementation is compatible with the S2R TC. More precisely, THOR is part of the R2R Offer Categorizer module (OC, see Section 3.1), which pre-analyzes travel offers before feeding them to THOR and retrieves the results of THOR's computation. The complete implementation of the OC is not part of this work.

2.2. Recommender Systems

Usually, an RS falls into one of the following categories [33,34]: (*i*) *Content-based*, where recommendations are provided based on the user's past purchases; (*ii*) *Collaborative*, where recommendations are based on other users with similar preferences; and (*iii*) *Hybrid*, which combine (*i*) and (*ii*).

Valliyammai et al. [35] propose a model-based, collaborative filtering system based on fuzzy c-means [36] for clustering and A-priori [37] for classification. Motivated by their future work, to enhance the proposed system, we utilize Support Vector Machines (SVM) for our proposed system.

Sebasti et al. [15] developed a framework that collects user profiles by requiring users to enter their details and general preferences and asking them to introduce their specific preferences for the current visit. Then, the system generates a list of activities that are likely of interest to the user. A hybrid RS classifies users into different categories—e.g., "Person with Children". Then, a content-based approach recommends more places according to the user's history, and a filter selects the proper places based on the current request. Unlike Sebasti et al. [15], our proposed approach relies only on the historical records of the travelers and does not require them to specify their preferences. In our system, a user can optionally specify some preferences which will be used by the recommender system. Further details are available in Section 3.

Table 1. Main features used for data generation along with their type and category.

Name	Type	Category	Name	Type	Category
Age	Cat.	Profile	Starting Point	Cat.	Offer
City	Cat.	Profile	Destination	Cat.	Offer
Country	Cat.	Profile	Via	List	Offer
Loyalty Cards	List	Profile	LegMode	List	Offer
Payment Cards	List	Profile	Class	List	Offer
PRM Type	List	Profile	Seats Type	List	Offer
Profile Type	Cat.	Profile	Arrival Time	Cat.	Offer
Quick	Float	Offer	Departure Time	Cat.	Offer
Reliable	Float	Offer	Preferred Transp. Types	List	Search
Cheap	Float	Offer	Preferred Carriers	List	Search
Comfortable	Float	Offer	Preferred Refund Type	Cat.	Search
Door-to-door	Float	Offer	Preferred Services	List	Search
Envir. Friendly	Float	Offer	Max. No. of Transfers	Int.	Search
Short	Float	Offer	Max. Transfers Duration	Cat.	Search
Multitasking	Float	Offer	Max. Walking Dist. to Stop	Cat.	Search
Social	Float	Offer	Walking Speed	Cat.	Search
Panoramic	Float	Offer	Cycling Distance to Stop	Cat.	Search
Healthy	Float	Offer	Cycling Speed	Cat.	Search
Legs Number	Int.	Offer	Driving Speed	Cat.	Search

Lorenzi et al. [38] decompose a recommendation into travel services, and different services are managed by different agents (that work cooperatively). As the recommendation process proceeds, each agent becomes expert in one or more specific travel service. The interaction among these specialized agents results in better recommendations, and an excellent offer will be generated by combining all services recommended by different agents in this method. Although the system could be suitable for one user, the performance can be quite different for different users. In other words, the system lacks personalization at the individual level.

Javadian et al. [39] propose a data-mining-based RS to rank offers. They use association rule mining to calculate the similarities between the user requests and the offers. The work first designs the features through a historical traveler database and, after a pre-processing phase, a knowledge base is set up by mining association rules from the database. Then, the knowledge base enables scoring each offer according to the characteristics of the user's mobility request. The main limitation of their work is the choice of rule-based algorithms (e.g., A priori) which are dependent on extracted/predefined rules which do not guarantee a good performance on the undefined cases. Moreover, like other systems introduced earlier, they rely only on the wisdom of the crowd and ignore the individual-level personalization.

Fang et al. [40] automatically generate, from a collection of documents based on Wikipedia and Twitter, temporal feature vectors of Point Of Interests (POIs) that include seasonal attractions such as water sports, snow festivals, and the viewing of scarlet maple leaves. Similarly, Coelho et al. [41] employ travel-related tweets to personalize recommendations regarding POIs for the user. Firstly, they categorize POIs as historical buildings, museums, parks, and restaurants; then, they build a classification model to classify the tweets. To obtain a better-personalized model, travel-related tweets of the user's friends and followers are also mined. The proposed systems by Fang et al. [40] and Coelho et al. [41] have two main limitations: they recommend POIs rather than complete travel offers for a journey, and they lack user modeling techniques.

Fararni et al. [42] explore a user profiling process according to the following scenarios: (*i*) *Inscription*: Inserting preferences, which is done directly by the user. (*ii*) *Social login*: Obtaining preferences from the user's Social Media (SM) accounts. (*iii*) *Consultation even without login*: Observation and analysis of the user behaviors. (*iv*) *Context*: Contextual information to generate dynamic visit schedules. The system also incorporates tourist services information, a contextual meta-model for analyzing the input data, a hybrid filtering process for storing the list of items with users' appreciation degree, and a trip planner for correlating all the choices as a trip. Our proposed system differs from the one by Fararni et al. [42] in many ways. Considering the *Inscription* scenario, although we incorporate user-provided preferences, our system does not depend on them. Concerning the users' behavioral aspects and context, our system implicitly learns the behavioral changes and contextual preferences by updating the users' preference models with recent purchase histories. Finally, concerning the *Social login* scenario, due to privacy concerns, we did not integrate SM as source of information in the current implementation of the system. Note that our proposed system can be extended to incorporate SM information using the social media core proposed in [9].

2.3. Travelers' Preferences

This section overviews the works analyzing the main characteristics of travels and travelers that potentially affect travelers' preferences. From their analysis, we derived a contextual dimension tree capturing the main features presented in [43].

Basile et al. [44] propose a system to understand if the users' preferences explicitly match their behavior. First, the system builds user profiles, i.e., general descriptions of the users based on their travel preferences; second, it builds profiles using evaluation data collected after the actual travels; and finally, the before- and after-travel user profiles are matched. This type of model can keep improving the accuracy of the computation of user preferences. Consonni et al. [45] collect travel-centered mobility data via crowdsourcing. The time spent on the travel is analyzed from a traveler's perspective: the user is asked which activities they have performed during the trip, and which factors have influenced their trip positively or negatively. After analyzing the data, the system can change the perspective on the travel time: instead of considering it simply as *spent* or *wasted*, the system characterizes the travel time in terms of the activities performed, i.e., *fitness*, *enjoyment*, and *productivity*. Boratto et al. [46] analyze and characterize user behavior during journey planning to get insights from different perspectives related to trip search options, i.e., both sorting and selection actions. While selecting different offers, the system can rapidly learn more about the users' preferences.

2.4. Cold Start Problem

The dependency on the existence of historical data is known as the "cold start problem" [47]. Work in [48–52] explores various techniques for determining the best item recommendations for a new user. These techniques employ strategies based on each item's popularity and/or the user profile. However, to provide the user with reliable recommendations, a content-based RS should have access to a sufficient number of user's records that allow it to determine the user's preferences. Therefore, a new user, having very few records, might not receive accurate recommendations. Moreover, although recommending a new user's top popular offers might increase the user's purchase likelihood, it decreases the personalization. Finally, collaborative methods can help to improve personalization, but the recommendations' precision might be quite low.

Clustering algorithms can be used to group users according to their profiles, and the resulting model can predict the cluster of a new user.

In our work, we design and build the RS block proposed by Javadian et al. [8,9]. To do so, we combine collaborative and content-based methods to develop a *hybrid* approach. Given a new user, we do not aim to find a single similar user, but we look for a group of users with similar characteristics instead. Then, we do not directly recommend to new users the travel offers that have been bought by similar groups. Rather, we use a content-based method to build the recommender model for the group and use the model to predict the new user's recommendations.

3. The Hybrid Offer Ranker (THOR)

This section details THOR's implementation and functions. THOR's source code is publicly available to researchers for testing and possible improvement (github.com/Ride2Rail/Learner_Ranker/tree/main/TravelOffer_RecommenderCore, accessed on 10 October 2022).

3.1. Overview

THOR aims to provide ranked travel offers to TC users according to their contextual preferences. Figure 1 provides a high-level overview of the THOR system.

Travel offers and user profiles are collected when TC users send mobility requests, where each request includes the so-called "search options"—i.e., situational preferences such as the desired means of transportation, the maximum number of connections, and so on (see also Table 1). In response, the Travel Solution Aggregator—a third-party application that queries TSPs for travel solutions—returns the list of offers to fulfill the mobility request. For each of the offers returned by the Travel Solution Aggregator, the Offer Categorizer module (OC) computes scores for the following travel categories: *fast*, *reliable*, *cheap*, *comfortable*, *door-to-door*, *environmentally friendly*, *short*, *multitasking*, *social*, *panoramic*, and *healthy* (the implementation of the OC module, developed within the R2R project, is not part of this work). The OC also extracts specific details of each leg of the travel offer (where a leg is a single piece of travel contained in an end-to-end journey), such as *transportation mode*, *seat type*, *carrier*, and *length*. In the rest of this work, we refer to the combination of category scores and leg-level information as the *enriched offer*. In turn, enriched offers are combined with the user's most recent profile information to build the *Ranker*'s input data. The Ranker uses the *personal, contextual preference* model of the user—elaborated by the *Learner*—to predict if they will buy the offer or not. Next, the ranked list of offers is shown to the user. After a travel offer is selected from the list, the user's historical data are updated with the offers in the list (where each offer is tagged as purchased/not purchased), and the Learner module uses the new records to update the user's contextual preference model.

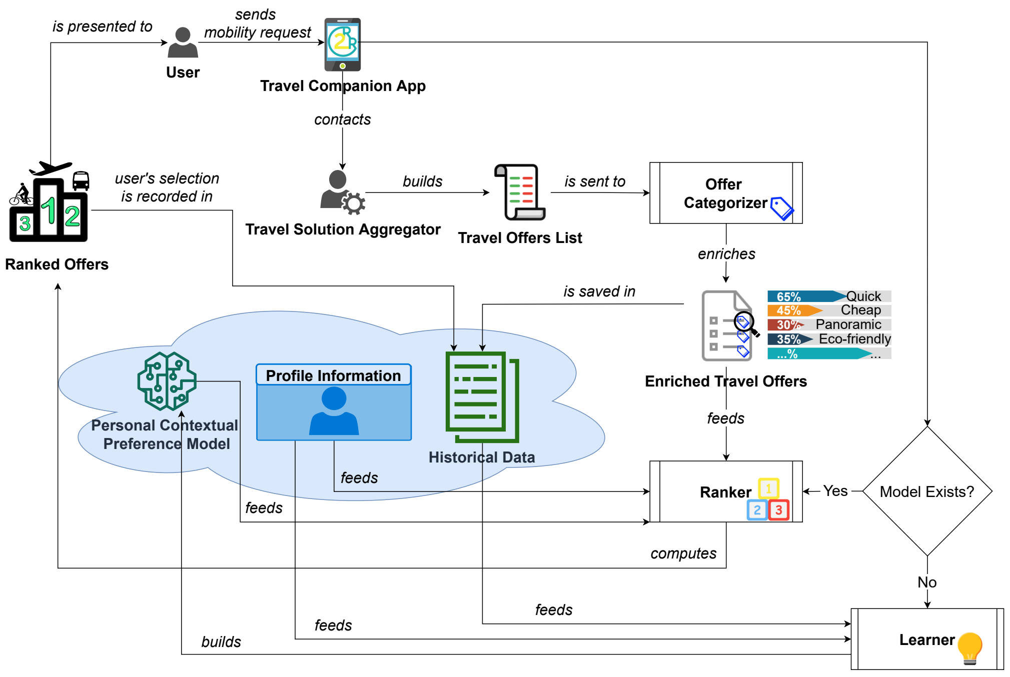

Figure 1. High-level representation of THOR.

We say that a user is *cold* if their historical database contains a number of records smaller than a given threshold (e.g., 100); among cold users, we have *new users*—i.e., users who just registered so the system does not have any historical records for them.

We separate *contextual preference models* into two categories: *user-specific recommender models*—i.e., preference models obtained using data related to a single user—and *cluster-wide recommender models*—i.e., models trained considering data from a set of travelers (instead of a single one) with similar profiles. We recognize users who have similar profiles by building a *cluster model*, which allows us to identify, for each user (even new ones), the cluster to which they belong.

3.2. Data Pre-Processing

In order to prepare the data for the learning and ranking modules, the framework applies the following pre-processing steps.

1. One-Hot Encoding:This step translates the raw textual data into a numerical one-hot version. Moreover, null data are replaced with zeros. Consider, for example, the feature "Profile", which takes one of the following four values: "Basic", "Business", "Family", or "Leisure". During one-hot encoding, "Profile" is split into four features (one for each possible value) that are mutually exclusive—i.e., only one of them can have a value equal to one, while the rest are zero.
2. Information-Less Columns Dropping (ILCD): in this step, the system deletes all columns that have the same value in the dataset. For instance, if the user has never changed their hometown, we can delete it because this feature means nothing to our system. Deleting these columns substantially speeds up the training phase.
3. Data Normalization: Since the scales and magnitudes of the features are not the same, if the original data values are used directly during the prediction phase, their degree of influence is different. The system applies a normalization process through which every feature will have the same influence on the result.

3.3. Learner Module

One of THOR's fundamental blocks is the Learner, which constantly updates users' preferences and builds the most recent contextual preference model for each user individually. Figure 2 details the logic control of the Learner module, and Algorithm 1 provides its pseudo-code.

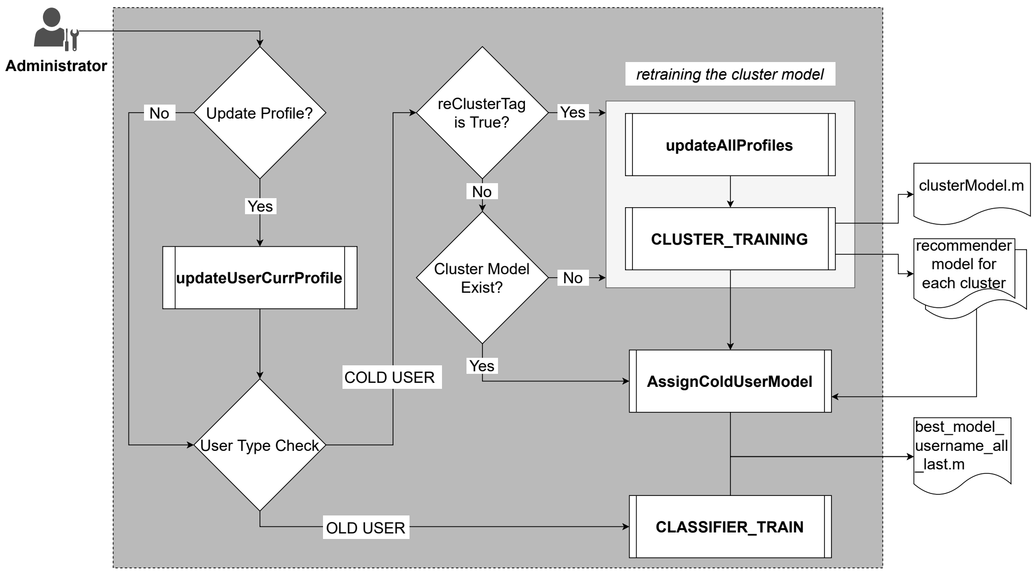

Figure 2. THOR's Training (Learning) Workflow.

An administrator can manually or automatically (according to a schedule) trigger the update of the user models. Before training the model, the administrator can choose to update the current user or not, read all the users' current profiles, and put them all into one table as the input data to the cluster training module. The administrator can also upload other users' profile data supplied by third parties to help the system train a better model at the first stage.

For a cold user, the administrator can re-train the cluster model by setting the value of `reClusterTag` to *true*, and the re-training process will also be completed when the system does not find the cluster model. During the cluster model training phase (which is encapsulated in algorithm `ClusterModelTrain` shown in Algorithm 2), all users' profiles are sent to `CLUSTER_TRAINING`; the system keeps track of the cluster model and of the cluster-wide recommender models. These models are then associated with the cold user based on the cluster to which the user belongs; in this way, we build the cold user's model by training it on the historical data of other—similar—travelers.

For an old user, the administrator uses the `CLASSIFIER_TRAIN` function to train or re-train the user's model and save the user's best model.

Algorithm 1 Learner

```
Input:  username; reClusterTag; new profile (optional).
Output: user-specific recommender model or cluster-wide recommender model.
 1: function ModelTrain(username, reClusterTag, new profile)
 2:     user's historical records ← fetch user records from database(username)
 3:     user type ← check the number of user's historical records
 4:
 5:     if user new profile is given then
 6:         update the database with new profile
 7:     end if
 8:
 9:     if user type is old user then
10:         user recommender model ← CLASSIFIER_TRAIN(user's historical records)
11:     else
12:         if cluster model does not exist OR reClusterTag is True then
13:             search ranges ← parameters range define
14:             cluster model, cluster-wide recommender models ← ClusterModelTrain(search
    ranges)
15:         end if
16:         user cluster ← get user cluster(cluster model, username)
17:         cluster-wide recommender model ← get cluster-wide recommender model(cluster-wide
    recommender models, user cluster)
18:         user recommender model ← copy model(cluster-wide recommender model)
19:     end if
20:     user-specific recommender model ← save model(user recommender model)
21: end function
```

Algorithm 2 Cluster Model Training

```
Input:  search ranges for each algorithm.
Output: cluster model; cluster-wide recommender models.
 1: function ClusterModelTrain(search ranges)
 2:     user profiles ← fetch all user profiles from the database
 3:     best parameters ← compute best parameters for algorithms(search ranges)
 4:     cluster model ← CLUSTER_TRAINING(best parameters, user profiles)
 5:
 6:     for cluster ∈ cluster model do
 7:         cluster historical records ← merge all the users' records in the cluster
 8:         cluster-wide recommender model ← CLASSIFIER_TRAIN(cluster historical records)
 9:     end for
10: end function
```

3.3.1. Cluster Training for New User

The pseudo-code for the training of the cluster-wide models that occur in case of a new user is presented in Algorithm 2. The system first computes the clusters of users, then, for each cluster, it learns a *cluster-wide recommender model* to solve the problem of recommendations for cold users. This module computes the best parameter setting and uses it to fit the cluster model. Then, it gathers the data for each cluster and trains its models. The function also supports using data supplied by the administrator themselves; otherwise, the system will automatically fetch all valid users' current profiles in the database.

Before fitting the model, the system finds the best parameters by using the parameter-tuning function. To do so, the first step is to define the search range for each parameter in different algorithms. In this work, we used two well-known clustering algorithms: *K-means* [24] and *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* [25] provided in *Sklearn.cluster* [53]. One of the main strengths of these algorithms is that they can be used easily with any data type, various distance functions, and efficient indexing approaches facilitating the analysis of large datasets [54]. To calculate the optimal number of clusters for *K-means* and DBSCAN, we employ the Elbow method and the Silhouette coefficient, respectively [55], which are briefly recalled here.

In particular, *K-means* aims to minimize the Sum of Squared Error (SSE) between the samples and the mass point of the cluster they belong to; the smaller the value of SSE, the tighter the clusters. Given a set of samples, the value of SSE typically decreases as the number of clusters increases, first slowly, then more steeply, until it again decreases slowly. This gives the SSE curve the shape of an "elbow", and the inflection point (i.e., the elbow) corresponds to the number of clusters that offers the best performance for the algorithm. The Elbow method allows us to determine the inflection point of the SSE curve, which corresponds to the optimal number of clusters to configure *K-means*.

To obtain more precise clustering results with DBSCAN, instead, we calculate the Silhouette coefficient. More precisely, the best coefficient is obtained when the distances among points in the same cluster (resp., different clusters)—i.e., the *cohesion* (resp., the *separation*)—are as small (resp., as big) as possible.

Comparing *K-means* and DBSCAN, although *K-means* requires a prototype-based concept of a cluster (i.e., the number of clusters and their initial centroids), it is useful for sparse and high dimensional data. On the other hand, DBSCAN is powerful in dealing with noises, although it does not perform very well for high dimensional data [56].

After obtaining the best parameters for the algorithms (e.g., the number of clusters), we use `CLUSTER_TRAINING` to compute the set of clusters of users with similar profiles. Next, for each cluster, we combine the historical records of all the users in the cluster and feed them to the `CLASSIFIER_TRAIN` module to build the cluster-wide recommender model.

Finally, when a new user registers to the system, THOR uses their profile information to compute the cluster to which the user belongs and associates the corresponding cluster-wide recommender model with the new user.

3.3.2. User Recommender Model Training

The `CLASSIFIER_TRAIN` function, whose pseudo-code is shown in Algorithm 3, is the core mechanism for building both user-specific and cluster-wide recommender models. More precisely, to train user-specific (resp., cluster-wide) recommender models, function `CLASSIFIER_TRAIN` receives as input the records of a single user (resp., of all users that belong to the cluster). The function first finds the best parameter setting, then uses it to build the recommender model.

To capture the user context, each user record is made of the most recent user profile information (*profile*), the enriched travel offer (*travel offer*, which includes the information whether it was purchased or not), and the search options (*request*) that were used in the mobility request to which the travel offers are the reply.

Algorithm 3 Recommender Model Training

```
Input:  user records, i.e., the most recent profile information (profile), purchased and not-
        purchased enriched travel offers (travel offers), as well as their corresponding search
        options (requests)
Output: recommender model.
 1: function CLASSIFIER_TRAIN(user records)
 2:     train data ← data preprocessing(user records)
 3:     search ranges ← parameters range define
 4:
 5:     for algo ∈ [KNN, SVC, DT, LR, RF] do
 6:         temp best model, score ← BSCV(algo, train data, search ranges)
 7:         best model ← compare scores of the models
 8:     end for
 9:     recommender model ← save the model in file(best model)
10: end function
```

To create the training dataset, the system takes the received user records and first performs a pre-processing step (see Section 3.2). Then, we use the data to find the best algorithm and best parameters and then save the final model.

Knowing whether each offer was bought or not allows the system to solve a classification problem, where each new travel offer is classified as “buy” or “not buy” depending on the current profile. More precisely, for each new offer, the system predicts the probability that the user will buy it and considers this value as the score of the offer; finally, offers are ranked according to their scores (i.e., the probability that they will be bought).

To compute the prediction, and in particular to build the personal recommender models, we use popular classification algorithms, all provided by the *Sklearn* package [53]: KNN [19], SVC [20], DT [21], RF [22], and LR [23]. The reason for choosing such algorithms instead of, say, neural networks is that they require much fewer data points, which is a crucial requirement in our case for building personal models [57].

Let us briefly point out the main advantages and disadvantages of each algorithm. KNN is non-parametric and does not make any prior assumption on the data distribution; however, it can exhibit poor performance for high dimensional data and in presence of irrelevant features. SVC is quite robust with respect to the behavior of observations that are far from the decision boundary, but it is not suitable for large data sets. Considering DT, although its results are interpretable, it is sometimes less accurate compared to the other algorithms. RF requires higher training time than other algorithms, but it is suitable when the dataset is large and interpretability is not a major concern. LR is very efficient to train and does not make any assumptions about distributions of classes in the feature space; however, if the number of records is much smaller than the number of features, it may result in overfitting [58].

The set of chosen algorithms covers many cases and situations (e.g., some algorithms work well with small datasets, others are better with big ones); THOR chooses the best-performing algorithm depending on the current training dataset, which ensures the quality of the recommendations. Therefore, a general evaluation method is needed to automatically compare the algorithms among them and find the best fit for the dataset. In this regard, the most popular one is Bayes Search Cross-Validation (BSCV) [59]. Grid Search Cross-Validation (GSCV) [60] is another appropriate method, especially for KNN; however, BSCV can be used for all the algorithms employed in this work. BSCV updates the current best model during each iteration and changes parameter settings according to the search ranges. If the parameter setting is invalid—i.e., the combination of values does not fit the algorithm—the system discards it and searches for another one. The best model for the current user, which depends on the best score given by BSCV, is then generated. Finally, we store the best model to be used in the future.

3.4. Ranker

The Ranker (shown in Algorithm 4) is the main component to get the recommendation results for a user. Its core is represented by function `CLASSIFIER_RESPONSE`, which, for each travel offer, computes the travel offer's score (i.e., the probability that the user will buy that travel offer) using the user's recommender model. The Ranker assumes that a recommender model has already been assigned to the user; if not, prior to the steps described in the following, it invokes the Learner block shown in Algorithm 1.

Initially, the Ranker receives the corresponding preference model of the user, i.e., the user-specific model in the case of the old user or the cluster-wide model in the case of a cold user. After getting the model, the Ranker uses function `CLASSIFIER_RESPONSE` to predict if the user will buy or not any of the offers and saves the results.

Algorithm 4 RANKER

Input: user's recommender model (model), most recent profile information (profile), search options (request), list of enriched travel offers (travel offers)
Output: ranked list of travel offers (ranked offers)

```
1: function RANKER(model, profile, request, travel offers)
2:     scored_travel_offers = []
3:
4:     for offer $\in$ travel offers do
5:         (offer_id, prediction's score) $\leftarrow$ CLASSIFIER_RESPONSE(model, profile, request, offer)
6:         scored_travel_offers.append((offer_id, prediction's score))
7:     end for
8:     ranked offers $\leftarrow$ sort(scored_travel_offers)
9: end function
```

More precisely, the `CLASSIFIER_RESPONSE` function preprocesses the input data and makes the predictions by using the user recommender model. Each travel offer in the set (which has been enriched by the OC) is joined with the user's current profile and with the information concerning the mobility request to form the raw data, which is fed as input to the prediction function. Using the recommender model associated with the user, we obtain, for each offer, the probability that it will be bought by the user and use that as the offer score.

Finally, the Ranker sorts the travel offers according to their scores and presents them to the user.

3.5. User Feedback

To have better accuracy for the recommendations, the system needs to update the user's historical records continuously. The collected user feedback consists of the purchasing decision after the recommendation: the user will typically buy an offer from the list shown to them and ignore the rest. However, if the system recommends too many offers to the user, part of them will not be seen by the user, and the model will be updated based only on the best recommendations. Hence, the top 30 recommended offers, plus the one bought by the user, are recorded in the historical dataset; this number may have little impact on a large dataset, but it may change the results a lot for a small dataset, especially for a new user.

4. Experimental Results

To evaluate the features of the THOR system, we carried out a few experiments with the two following goals. Since, as mentioned in Section 1, to the best of our knowledge in the literature, there are no other works directly comparable to THOR, a direct comparison of the performance of THOR with that of similar tools is not possible. However, as described in Section 4.1, we first aim to quantitatively evaluate the accuracy of THOR (and in particular of its classifiers) when predicting the category (bought/not bought) of each travel offer received. Second (Section 4.2), we evaluate its ability to suitably rank the sets of offers received, in particular when the user context (especially the user profile) changes. Since there are no suitable metrics to assess the ranking mechanism in a quantitative manner (e.g., through a notion of accuracy) [61], we designed a controlled experiment to evaluate the quality of the rankings instead.

4.1. Validating the Classifiers

To test THOR, we need an existing dataset. At first, we attempted to find a suitable publicly available dataset for this purpose; although the available datasets had some information about travelers in Europe and public transport facilities, none of them could match (even partially) with the Shift2Rail's data structure. To do so, we designed a data generation pipeline using some rules to avoid having a completely random dataset. An advantage of this line of work is that knowing the distribution of the dataset allows us to validate the results. Table 1 provides the main features that we used during our experiment. "Profile" encompasses a set of features such as age and list of loyalty cards saved in the user's profile. "Search Options (Search)" includes a set of features related to the submitted mobility request by the user (e.g., preferred transportation type). Lastly, "Travel Offer (Offer)" is the set of features extracted from the offer to be ranked (e.g., number of legs). We generated a dataset with 1000 unique user profiles and a total of 101,028 records (approximately 100 records—i.e., travel offers—per user, so each user has enough records to be considered "old", and a user-specific recommender model can be trained for them).

The dataset was generated randomly, but we introduced a few rules to avoid making it uniform and to create "hidden patterns" to check whether our recommendation mechanism was able to pick them up. For example, in the generated dataset, 40% of the travel offers associated with users who are Persons With Reduced Mobility (PRM) have a single leg (i.e., it holds that "Legs Number = 1"), and 80% are *short*. Then, we associated with each offer a "Bought" tag that depended on one of the features of the offers (e.g., "short"), and we finally randomly changed 10% of the *bought* (resp., *not bought*) travel offers to *not bought* (resp., *bought*). In this way, the *bought* travel offers are not evenly distributed across the dataset. We use 80% of the records in the dataset for training the classifiers and 20% for testing their accuracy. A perfect classifier would be able to correctly guess which, among the 20% travel offers used for testing are *bought* and which are not.

Figure 3 reports the box plot of the time required, for each user, to train each algorithm (training was carried out on a MacBook Pro with CPU Core i9 and 16 GB RAM). If we consider, for each user, the cumulative time that it takes to train all algorithms (i.e., the time to train KNN on the user records, plus the time to train SVC on the same user records, and so on), then the sum of the average training times (i.e., the values highlighted as yellow lines in Figure 3) is around 1 s, and the sum of the maximum required training times (i.e., the highest dots in Figure 3) is close to 2.1 s. Moreover, there exists a minor computational cost (a few milliseconds) to retrieve the learned model and predict the scores of the travel offers.

After getting all the best models of the test users—i.e., the combination of parameters for the algorithm with the best performance—we recorded the accuracy value of each algorithm. The accuracy is obtained by considering true positives (resp., true negatives) as the travel offers that are tagged as "purchased" (resp., "not purchased") in the dataset and for which THOR returned a correct prediction of 1 (resp., 0) by computing the following quantity:

$$\frac{\textit{true positives} + \textit{true negatives}}{\textit{number of records}} \times 100$$

Figure 4 details the accuracies of each algorithm for all the users. The last box plot provides the scores obtained from the best algorithm for each user. We optimize the model by selecting the best algorithm automatically. The average accuracy is equal to 91%; the highest accuracy is equal to 100%, while the lowest accuracy is equal to 72%.

Figure 5 shows the proportion of the models obtained by each algorithm as the optimal model. We can see that LR is the most suitable and KNN and RF are the least suitable algorithms for our test data.

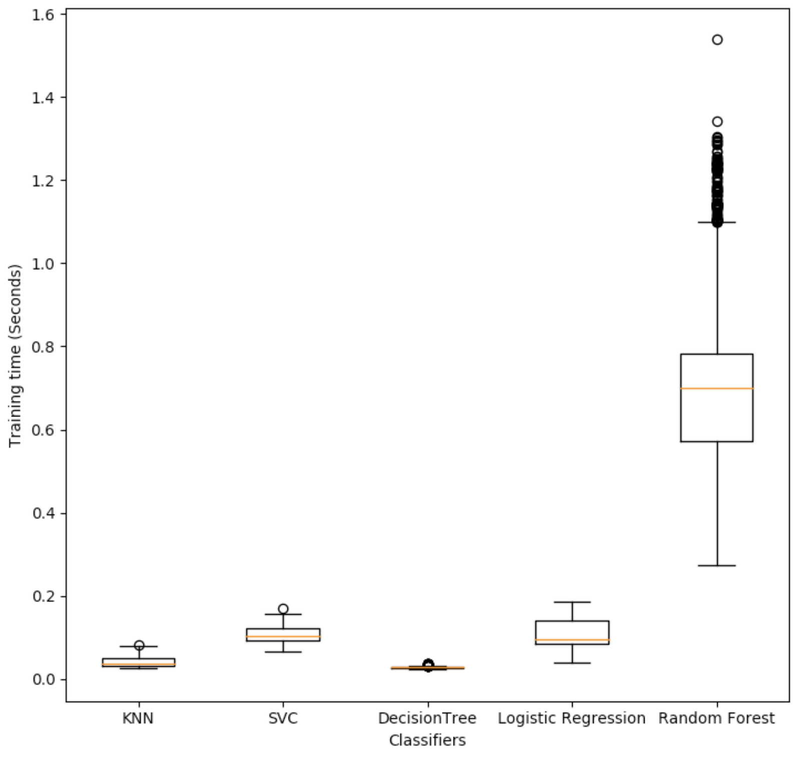

Figure 3. The classifiers' training time for all the users.

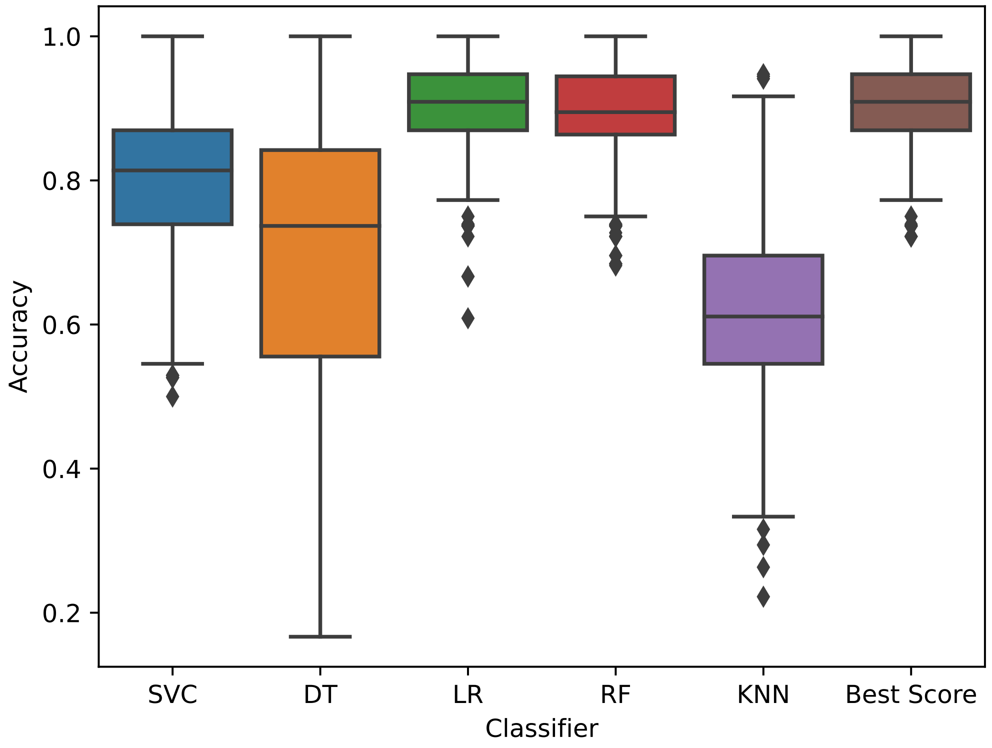

Figure 4. The accuracies of each algorithm for all the users. The last box plot provides the scores obtained from the best algorithm for each user.

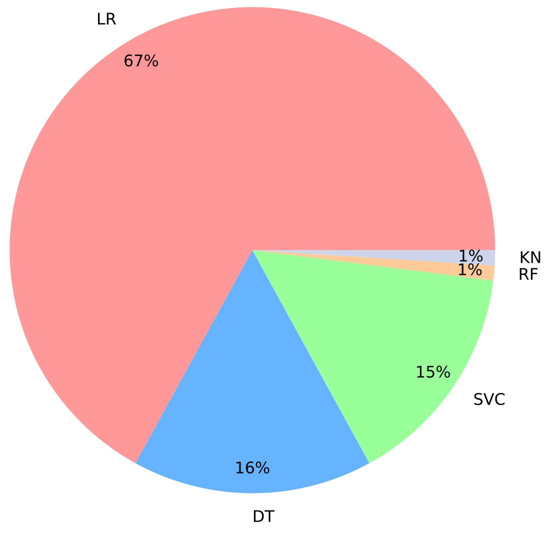

Figure 5. The probability distribution of different algorithms is used as the best model.

4.2. Validating the Ranker

The procedure discussed in Section 4.1 focuses on the validation of the performance of the classifiers, and it is not well suited to validate the proposed ranking mechanism. Therefore, in this section, we present a scenario-based validation procedure to evaluate how the system ranks the travel offers for a specific user depending on the user context. To do so, we manually created a data set for a fictional person named Sarah. The dataset is such that Sarah selects different types of travel offers depending on her context. We expect that THOR detects contextual preferences of Sarah and ranks the travel offers accordingly.

Sarah is a 35-year-old assistant professor at the Polytechnic University of Atlantis (EU). She lives in Atlantis, and due to the research projects she is working on, she frequently travels to many European countries to meet the project's partners. She has been using the TC application for some time to find the most suitable travel offers; therefore, the system has some records of her choices in different contexts. Sarah's travel records can be divided into two typical situations. The first set of records concerns a period in which she is healthy—i.e., she does not have any issues that could make her a PRM. In these cases, she typically selects business class trips, window seats, does not put any constraints (in the search options) on transfer duration and number of stops, and so on. Since she cares a lot about global warming, she prioritizes travel offers with the minimum carbon footprint—i.e., those with the highest environmentally-friendly score even if they are not cheap or are not quick—and this results in her not choosing private taxis and similar options with low environmentally-friendly scores. Moreover, she likes travel offers that have high panoramic scores; therefore, she mostly avoids travel offers that cannot fulfill this preference (e.g., underground transportation). In addition, this results in Sarah choosing mostly travel offers with many legs (changes), low door-to-door scores, and sometimes low comfort scores.

The second set of records is related to a period in which she needed to use a wheelchair due to a car accident. As a result, she updated her profile information to mention her new PRM status, and she started favoring, in her selections, travel offers with characteristics such as door-to-door, comfortable, quick, and so on, even though they are totally different from what she used to choose. As a result, her personal recommender model is updated automatically using the new profile information. For instance, to satisfy the door-to-door requirement, as a PRM, she always chooses travel offers that include taxis on the first and last legs of the trip. Other changes in her selections related to modes of transportation include not choosing ship and bus trips, which she used to choose before. Moreover, if available, she chooses large seats. In other words, Sarah selects offers that are more suitable for PRM people. These offers are now part of her user records, where each record includes the profile information under which the choices were made.

Using these records (first and second sets), we trained a personal recommender model (using again 80% of the records for training and 20% for testing) that, on average, showed 90% accuracy when predicting travel offers that Sarah will buy. This shows that even when the context of the user—hence their patterns of behavior—changes, if the profile information is suitably updated, the system is able to adapt to these patterns and ranks travel offers according to the user's contextual preferences.

Since the purpose of this experiment is to test the ranking mechanism, we assumed a situation in which Sarah becomes healthy again and does not use a wheelchair anymore. For a given mobility request at this time, we generated three potential travel offers: travel offer A, with characteristics most similar to the time when she was not a PRM, travel offer B, with the characteristics most similar to the time when she was a PRM, and travel offer C, with blended characteristics.

As expected, the Ranker could successfully rank travel offer A as the first, C as the second, and B as the third.

5. Conclusions and Future Work

In this work, we designed and implemented The Hybrid Offer Ranker (THOR), a personalized, context-aware, hybrid recommender system that employs various state-of-the-art classification algorithms (DT, KNN, LR, RF, and SVC) to tackle RQ1—i.e., to learn the travelers' contextual preferences and rank travel offers accordingly. Moreover, to address RQ2, we used the *K*-means and DBSCAN clustering algorithms to deal with the cold-start problem for new users. To tune the algorithms' hyper-parameters, we designed a grid search (GSCV) mechanism which finds the set of hyper-parameters automatically. THOR keeps learning as soon as a new record or user is registered in the system, thus, keeping the recommender models up-to-date. Notice that the modular design of THOR allows the integration of classification and clustering algorithms other than those used in this work.

Since the TC application is under development, there exists no real data for testing purposes. Hence, to test the performance of THOR, we automatically synthesized a dataset of 1000 unique user profiles and more than 100,000 travel offers, and we also manually built a controlled dataset for a specific user according to a predefined scenario. On both datasets, THOR showed an accuracy higher than 90%. These are promising results that show that THOR can be integrated with other TC modules to be tested in demo sites.

In the future, we plan to extend/improve THOR in several directions. As soon as we acquire enough real data, we plan to test the performance of THOR by using various feature selection methods [62–64] which potentially might result in reducing the complexity of the model while improving its accuracy.

Additionally, we plan to use deep learning approaches, such as the multimodal deep learning methods presented in [65–67], for training the cluster-wide recommender models. In addition, various transfer learning methods [68] could be exploited to reduce the training time while updating the cluster-wide recommender models.

We plan to build the social media core proposed in [9] as a tool [69,70] to characterize urban mobility patterns [71,72]. Moreover, the social media core will enable the system's stakeholders to understand user preferences during online events [73,74] which bring many travelers to specific European cities. Consequently, we plan to design predictive models as proposed in [75] to predict the popularity of online content generated by the stakeholders to maximize the visibility and popularity of their news and advertisements. Last but not least, the social media core will enable us to extract the conversation graphs [76,77] around specific topics, build conversational agents [78], and facilitate customer relationship management.

Author Contributions: A.J.S.: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing—Original Draft, Writing—Review & Editing, Visualization. M.S.: Conceptualization, Methodology, Software, Validation, Investigation, Data Curation, Writing—Original Draft, Writing—Review & Editing, Visualization. C.G.: Methodology, Software, Validation, Investigation, Data Curation, Writing—Original Draft, Writing—Review & Editing, Visualization. M.R.: Supervision, Funding acquisition, Conceptualization, Methodology, Validation, Investigation, Writing—Review & Editing. F.S.: Supervision, Conceptualization, Methodology, Validation, Investigation, Writing—Review & Editing. L.T.: Supervision, Conceptualization, Methodology, Validation, Investigation, Writing—Review & Editing. All authors have read and agreed to the published version of the manuscript.

Funding: Work supported by Shift2Rail and the EU H2020 research and innovation programme under grant agreement No: 881825 (RIDE2RAIL).

Data Availability Statement: THOR's source code and the generated datasets are publicly available at: https://github.com/Ride2Rail/Learner_Ranker/tree/main/TravelOffer_RecommenderCore. (accessed on 10 October 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brambilla, M.; Javadian Sabet, A.; Masciadri, A. Data-driven user profiling for smart ecosystems. In *Smart Living between Cultures and Practices. A Design Oriented Perspective*; Mandragora: Milan, Italy, 2019; pp. 84–98.
2. Ferreira, J.C.; Martins, A.L.; da Silva, J.V.; Almeida, J. T2*—Personalized Trip Planner. In Proceedings of the Ambient Intelligence–Software and Applications—8th International Symposium on Ambient Intelligence (ISAmI 2017), Porto, Portugal, 21–23 June 2017; De Paz, J.F.; Julián, V.; Villarrubia, G.; Marreiros, G.; Novais, P., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 167–175.
3. Chen, X.; Liu, Q.; Qiao, X. Approaching Another Tourism Recommender. In Proceedings of the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 11–14 December 2020; pp. 556–562. [CrossRef]
4. Ricci, F.; Rokach, L.; Shapira, B. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2011; pp. 1–35. [CrossRef]
5. Adomavicius, G.; Tuzhilin, A. Context-Aware Recommender Systems. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2011; pp. 217–253. [CrossRef]
6. Abbar, S.; Bouzeghoub, M.; Lopez, S. Context-aware recommender systems: A service-oriented approach. In Proceedings of the VLDB PersDB Workshop, Lyon, France, 28 August 2009; pp. 1–6.
7. Villegas, N.M.; Sánchez, C.; Díaz-Cely, J.; Tamura, G. Characterizing context-aware recommender systems: A systematic literature review. *Knowl.-Based Syst.* **2018**, *140*, 173–200. [CrossRef]
8. Sabet, A.J.; Rossi, M.; Schreiber, F.A.; Tanca, L. Context Awareness in the Travel Companion of the Shift2Rail Initiative. In Proceedings of the 28th Italian Symposium on Advanced Database Systems, CEUR-WS.org, CEUR Workshop Proceedings, Villasimius, Sardinia, Italy, 21–24 June 2020; Volume 2646, pp. 202–209.
9. Javadian Sabet, A.; Rossi, M.; Schreiber, F.A.; Tanca, L. Towards Learning Travelers' Preferences in a Context-Aware Fashion. In *Proceedings of the Ambient Intelligence—Software and Applications*; Springer International Publishing: Cham, Switzerland, 2021; pp. 203–212. [CrossRef]
10. Dey, A. Understanding and Using Context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7. [CrossRef]
11. Bolchini, C.; Curino, C.A.; Quintarelli, E.; Schreiber, F.A.; Tanca, L. A data-oriented survey of context models. *ACM Sigmod Rec.* **2007**, *36*, 19–26. [CrossRef]
12. Alegre, U.; Augusto, J.C.; Clark, T. Engineering context-aware systems and applications: A survey. *J. Syst. Softw.* **2016**, *117*, 55–83. [CrossRef]
13. Hong, J.Y.; Suh, E.H.; Kim, S.J. Context-aware systems: A literature review and classification. *Expert Syst. Appl.* **2009**, *36*, 8509–8522. [CrossRef]
14. Ng, P.C.; She, J.; Cheung, M.; Cebulla, A. An Images-Textual Hybrid Recommender System for Vacation Rental. In Proceedings of the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), Taipei, Taiwan, 20–22 April 2016; pp. 60–63. [CrossRef]
15. Sebastia, L.; Garcia, I.; Onaindia, E.; Guzman, C. e-Tourism: A Tourist Recommendation and Planning Application. In Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence, Dayton, OH, USA, 3–5 November 2008; Volume 2, pp. 89–96. [CrossRef]
16. Kbaier, M.E.B.H.; Masri, H.; Krichen, S. A Personalized Hybrid Tourism Recommender System. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; pp. 244–250. [CrossRef]

17. Bolchini, C.; Quintarelli, E.; Tanca, L. CARVE: Context-aware automatic view definition over relational databases. *Inf. Syst.* **2013**, *38*, 45–67. [CrossRef]
18. Arnaoutaki, K.; Bothos, E.; Magoutas, B.; Aba, A.; Esztergár-Kiss, D.; Mentzas, G. A Recommender System for Mobility-as-a-Service Plans Selection. *Sustainability* **2021**, *13*, 8245. [CrossRef]
19. Jaafar, H.B.; Mukahar, N.B.; Binti Ramli, D.A. A methodology of nearest neighbor: Design and comparison of biometric image database. In Proceedings of the 2016 IEEE Student Conference on Research and Development (SCOReD), Kuala Lumpur, Malaysia, 30 August 2016; pp. 1–6. [CrossRef]
20. Lan, L.S. M-SVC (mixed-norm SVC)—A novel form of support vector classifier. In Proceedings of the 2006 IEEE International Symposium on Circuits and Systems, Kos, Greece, 21–24 May 2006; p. 3264. [CrossRef]
21. Yu, Y.; Fu, Z.-L.; Zhao, X.-H.; Cheng, W.-F. Combining Classifier Based on Decision Tree. In Proceedings of the 2009 WASE International Conference on Information Engineering, Taiyuan, China, 10–11 July 2009; Volume 2, pp. 37–40. [CrossRef]
22. Bingzhen, Z.; Xiaoming, Q.; Hemeng, Y.; Zhubo, Z. A Random Forest Classification Model for Transmission Line Image Processing. In Proceedings of the 2020 15th International Conference on Computer Science Education (ICCSE), Delft, The Netherlands, 18–22 August 2020; pp. 613–617. [CrossRef]
23. Zou, X.; Hu, Y.; Tian, Z.; Shen, K. Logistic Regression Model Optimization and Case Analysis. In Proceedings of the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 19–20 October 2019; pp. 135–139. [CrossRef]
24. Lu, S.; Yu, H.; Wang, X.; Zhang, Q.; Li, F.; Liu, Z.; Ning, F. Clustering Method of Raw Meal Composition Based on PCA and Kmeans. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 9007–9010. [CrossRef]
25. Smiti, A.; Elouedi, Z. DBSCAN-GM: An improved clustering method based on Gaussian Means and DBSCAN techniques. In Proceedings of the 2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES), Lisbon, Portugal, 13–15 June 2012; pp. 573–578. [CrossRef]
26. Golightly, D.; Comerio, M.; Consonni, C.; Vaghi, C.; Pistilli, G.; Rizzi, G.; Di Pasquale, G.; Palacin, R.; Boratto, L.; Scrocca, M. Ride2Rail: Integrating ridesharing for attractive multimodal rail journeys. In Proceedings of the World Congress Rail Research 2022, Birmingham, UK, 6–10 June 2022.
27. Sadeghi, M.; Buchníček, P.; Carenini, A.; Corcho, O.; Gogos, S.; Rossi, M.; Santoro, R. SPRINT: Semantics for PerfoRmant and scalable INteroperability of multimodal Transport. In Proceedings of the TRA, Helsinki, Finland, 27–30 April 2020; pp. 1–10.
28. Hosseini, M.; Kalwar, S.; Rossi, M.G.; Sadeghi, M. Automated mapping for semantic-based conversion of transportation data formats. In Proceedings of the 1st International Workshop On Semantics For Transport, Karlsruhe, Germany, 9 September 2019; Volume 2447, pp. 1–6.
29. Kalwar, S.; Sadeghi, M.; Javadian Sabet, A.; Nemirovskiy, A.; Rossi, M.G. SMART: Towards Automated Mapping between Data Specifications. In Proceedings of the 33rd International Conference on Software Engineering and Knowledge Engineering, SEKE 2021, KSIR Virtual Conference Center, Pittsburgh, PA, USA, 1–10 July 2021. [CrossRef]
30. Carenini, A.; Dell'Arciprete, U.; Gogos, S.; Kallehbasti, M.M.P.; Rossi, M.; Santoro, R. ST4RT—Semantic Transformations for Rail Transportation. In Proceedings of the TRA 2018, Vienna, Austria, 16–19 April 2018; pp. 1–10.
31. Alobaid, A.; Garijo, D.; Poveda-Villalón, M.; Santana-Perez, I.; Fernández-Izquierdo, A.; Corcho, O. Automating ontology engineering support activities with OnToology. *J. Web Semant.* **2019**, *57*, 100472. [CrossRef]
32. Sadeghi, M.; Sartor, L.; Rossi, M. A Semantic-Based Access Control Mechanism for Distributed Systems. In Proceedings of the 36th Annual ACM Symposium on Applied Computing, SAC' 21, Gwangju, Korea, 22–26 March 2021; Association for Computing Machinery, New York, NY, USA, 2021; pp. 1864–1873. [CrossRef]
33. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
34. Cai, Y.; Leung, H.F.; Li, Q.; Min, H.; Tang, J.; Li, J. Typicality-Based Collaborative Filtering Recommendation. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 766–779. [CrossRef]
35. Valliyammai, C.; PrasannaVenkatesh, R.; Vennila, C.; Krishnan, S.G. An intelligent personalized recommendation for travel group planning based on reviews. In Proceedings of the 2016 Eighth International Conference on Advanced Computing (ICoAC), Chennai, India, 19–21 January 2017; pp. 67–71. [CrossRef]
36. Cao, Y.; Li, Y. An intelligent fuzzy-based recommendation system for consumer electronic products. *Expert Syst. Appl.* **2007**, *33*, 230–240. [CrossRef]
37. Agrawal, R.; Srikant, R. Fast algorithms for mining association rules. In Proceedings of the 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, 12–15 September 1994; Volume 1215, pp. 487–499.
38. Lorenzi, F.; Loh, S.; Abel, M. PersonalTour: A Recommender System for Travel Packages. In Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Lyon, France, 22–27 August 2011; Volume 2, pp. 333–336. [CrossRef]
39. Sabet, A.J.; Gopalakrishnan, S.; Rossi, M.; Schreiber, F.A.; Tanca, L. Preference Mining in the Travel Domain. In Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; pp. 358–365. [CrossRef]

40. Fang, G.S.; Kamei, S.; Fujita, S. Automatic Generation of Temporal Feature Vectors with Application to Tourism Recommender Systems. In Proceedings of the 2016 Fourth International Symposium on Computing and Networking (CANDAR), Hiroshima, Japan, 22–25 November 2016; pp. 676–680. [CrossRef]
41. Coelho, J.; Nitu, P.; Madiraju, P. A Personalized Travel Recommendation System Using Social Media Analysis. In Proceedings of the 2018 IEEE International Congress on Big Data (BigData Congress), Seattle, DC, USA, 10–13 December 2018; pp. 260–263. [CrossRef]
42. Fararni, K.A.; Nafis, F.; Aghoutane, B.; Yahyaouy, A.; Riffi, J.; Sabri, A. Hybrid recommender system for tourism based on big data and AI: A conceptual framework. *Big Data Min. Anal.* **2021**, *4*, 47–55. [CrossRef]
43. Shekari, M.; Sabet, A.J.; Guan, C.; Rossi, M.; Schreiber, F.A.; Tanca, L. Personalized Context-Aware Recommender System for Travelers. In Proceedings of the 30th Italian Symposium on Advanced Database Systems, SEBD 2022, Tirrenia, Italy, 19–22 June 2022; Amato, G., Bartalesi, V., Bianchini, D., Gennaro, C., Torlone, R., Eds.; 2022, Volume 3194, pp. 497–504.
44. Basile, S.; Consonni, C.; Manca, M.; Boratto, L. Matching User Preferences and Behavior for Mobility. In Proceedings of the 31st ACM Conference on Hypertext and Social Media, HT '20, Online, 13–15 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 141–150. [CrossRef]
45. Consonni, C.; Basile, S.; Manca, M.; Boratto, L.; Freitas, A.; Kovacikova, T.; Pourhashem, G.; Cornet, Y. What's Your Value of Travel Time? Collecting Traveler-Centered Mobility Data via Crowdsourcing. *arXiv* **2021**, arXiv:cs.CY/2104.05809.
46. Boratto, L.; Manca, M.; Lugano, G.; Gogola, M. Characterizing user behavior in journey planning. *Computing* **2020**, *102*. [CrossRef]
47. Schein, A.I.; Popescul, A.; Ungar, L.H.; Pennock, D.M. Methods and metrics for cold-start recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in IR, Tampere, Finland, 11–15 August 2002; pp. 253–260.
48. Rashid, A.M.; Albert, I.; Cosley, D.; Lam, S.K.; McNee, S.M.; Konstan, J.A.; Riedl, J. Getting to Know You: Learning New User Preferences in Recommender Systems. In Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI '02, College Station, TX, USA, 21–25 March 2022; Association for Computing Machinery: New York, NY, USA, 2002; pp. 127–134. [CrossRef]
49. Guo, G. Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In Proceedings of the 7th ACM conference on Recommender Systems, Hong Kong, 12–16 October 2013; pp. 451–454.
50. Yu, K.; Schwaighofer, A.; Tresp, V.; Xu, X.; Kriegel, H.P. Probabilistic memory-based collaborative filtering. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 56–69. [CrossRef]
51. Ghodsad, P.R.; Chatur, P.N. Handling User Cold-Start Problem for Group Recommender System Using Social Behaviour Wise Group Detection Method. In Proceedings of the 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE), San Salvador, El Salvador, 22–24 August 2018; pp. 1–5. [CrossRef]
52. Sang, A.; Vishwakarma, S.K. A ranking based recommender system for cold start data sparsity problem. In Proceedings of the 2017 Tenth International Conference on Contemporary Computing (IC3), Noida, India, 10–12 August 2017; pp. 1–3. [CrossRef]
53. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
54. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.* **2017**, *42*, 3068335. [CrossRef]
55. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Berlin, Germany, 2013; Volume 112.
56. Kanagala, H.K.; Jaya Rama Krishnaiah, V. A comparative study of K-Means, DBSCAN and OPTICS. In Proceedings of the 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 7–9 January 2016; pp. 1–6. [CrossRef]
57. Kumar, R.; Verma, R. Classification algorithms for data mining: A survey. *Int. J. Innov. Eng. Technol. (Ijiet)* **2012**, *1*, 7–14.
58. Narayanan, U.; Unnikrishnan, A.; Paul, V.; Joseph, S. A survey on various supervised classification algorithms. In Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 1–2 August 2017; pp. 2118–2124. [CrossRef]
59. Chung, T.H.; Burdick, J.W. Analysis of Search Decision Making Using Probabilistic Search Strategies. *IEEE Trans. Robot.* **2012**, *28*, 132–144. [CrossRef]
60. Huang, Q.; Mao, J.; Liu, Y. An improved grid search algorithm of SVR parameters optimization. In Proceedings of the 2012 IEEE 14th International Conference on Communication Technology, Chengdu, China, 19–21 October 2012; pp. 1022–1026. [CrossRef]
61. Shani, G.; Gunawardana, A. Evaluating recommendation systems. In *Recommender Systems Handbook*; Springer: Berlin, Germany, 2011; pp. 257–297.
62. Hosseini, M. Feature Selection for Microarray Classification Problems. Master's Thesis, Politecnico di Milano, Milan, Italy, 2018.
63. Brankovic, A.; Hosseini, M.; Piroddi, L. A Distributed Feature Selection Algorithm Based on Distance Correlation with an Application to Microarrays. *ACM Trans. Comput. Biol. Bioinform.* **2019**, *16*, 1802–1815. [CrossRef]
64. Rajeswari, K. Feature selection by mining optimized association rules based on apriori algorithm. *Int. J. Comput. Appl.* **2015**, *119*, 30–34. [CrossRef]
65. Hong, D.; Gao, L.; Yokoya, N.; Yao, J.; Chanussot, J.; Du, Q.; Zhang, B. More Diverse Means Better: Multimodal Deep Learning Meets Remote-Sensing Imagery Classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 4340–4354. [CrossRef]

66. Hong, D.; Gao, L.; Yao, J.; Zhang, B.; Plaza, A.; Chanussot, J. Graph Convolutional Networks for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 5966–5978. [CrossRef]
67. Wu, X.; Hong, D.; Chanussot, J. Convolutional Neural Networks for Multimodal Remote Sensing Data Classification. *IEEE Trans. Geosci. Remote. Sens.* **2022**, *60*, 1–10. [CrossRef]
68. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
69. Cea-Morán, J.J.; González-Briones, A.; De La Prieta, F.; Prat-Pérez, A.; Prieto, J. Extraction of Travellers' Preferences Using Their Tweets. In *Proceedings of the International Symposium on Ambient Intelligence*; Springer: Berlin, Germany, 2020; pp. 224–235.
70. Rivas, A.; González-Briones, A.; Cea-Morán, J.J.; Prat-Pérez, A.; Corchado, J.M. My-Trac: System for Recommendation of Points of Interest on the Basis of Twitter Profiles. *Electronics* **2021**, *10*, 1263. [CrossRef]
71. Manca, M.; Boratto, L.; Morell Roman, V.; Martori i Gallissà, O.; Kaltenbrunner, A. Using social media to characterize urban mobility patterns: State-of-the-art survey and case-study. *Online Soc. Netw. Media* **2017**, *1*, 56–69. [CrossRef]
72. Balduini, M.; Brambilla, M.; Della Valle, E.; Marazzi, C.; Arabghalizi, T.; Rahdari, B.; Vescovi, M. Models and Practices in Urban Data Science at Scale. *Big Data Res.* **2019**, *17*, 66–84. [CrossRef]
73. Javadian Sabet, A. Social Media Posts Popularity Prediction during Long-Running Live Events. A Case Study on Fashion Week. Master's Thesis, Politecnico di Milano, Milan, Italy, 2019.
74. Brambilla, M.; Javadian Sabet, A.; Hosseini, M. The role of social media in long-running live events: The case of the Big Four fashion weeks dataset. *Data Brief* **2021**, *35*, 106840. [CrossRef]
75. Javadian Sabet, A.; Brambilla, M.; Hosseini, M. A multi-perspective approach for analyzing long-running live events on social media: A case study on the "Big Four" international fashion weeks. *Online Soc. Netw. Media* **2021**, *24*, 100140. [CrossRef]
76. Brambilla, M.; Javadian, A.; Sulistiawati, A.E. Conversation Graphs in Online Social Media. In Proceedings of the Web Engineering, ICWE 2021, Biarritz, France, 18–21 May 2021; Springer International Publishing: Cham, Switzerland, 2021; pp. 97–112. [CrossRef]
77. Brambilla, M.; Javadian Sabet, A.; Kharmale, K.; Sulistiawati, A.E. Graph-Based Conversation Analysis in Social Media. *Big Data Cogn. Comput.* **2022**, *6*, 113. [CrossRef]
78. Scotti, V.; Tedesco, R.; Sbattella, L. A Modular Data-Driven Architecture for Empathetic Conversational Agents. In Proceedings of the 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Korea, 17–20 January 2021; pp. 365–368. [CrossRef]