POLITECNICO DI MILANO
School of Industrial and Information Engineering
Department of Electronics, Information and Bioengineering
Master of Science in Computer Engineering

Customer Preference Mining in the Travel Companion of the Shift2Rail Initiative

Supervisor: Prof. Letizia Tanca
Co-Supervisors: Prof. Matteo Rossi
Prof. Fabio A. Schreiber
Eng. Alireza Javadian Sabet

Candidate:
Sankari Gopalakrishnan
10641354/914809

Academic Year 2019-2020

Acknowledgement

I would like to convey my sincere gratitude to my supervisor Prof. Letizia Tanca for giving me the oppurtunity and the continuous support for my thesis. I would also like to thank Prof. Matteo Rossi and Prof. Fabio A. Schreiber for their encouragement and insightful suggestions and Eng. Alireza Javadian Sabet for his guidance throughout my research.

Amidst the COVID-19 pandemic, the team helped me connect online and were always present to clarify my doubts and make continuous progress.

I would like to take this opportunity to thank my partner for being the constant support system encouraging me to achieve my goals in life.

Thanks to all my friends and family who have stood by me at all times giving me strength

This has been a special journey and the memories will be treasured.

Contents

List of Abbreviations

CARVE	*Context-aware automatic view definition over relational databases*
CDT	*Context Dimension Tree*
ER Diagram	*Entity Relationship Diagram*
R2R	*Ride2Rail initiative*
RS	*Recommender Systems*
S2R	*Shift2Rail initiative*
TC	*Travel Companion*

List of Figures

List of Tables

Abstract

Personalisation of user experience through recommendations involves understanding their preferences and the context they are living in. In this work, we aim to present a set of ranked travel offers as an answer to a travel request made by a user. In order to give a sensible answer, it is important to learn the users' preferences over time, and use this information to understand the traveller's needs. Our solution is based on a data-mining-based recommender system. We start by designing the features of the application by designing a database of historical traveller data and populating it with a set of rules that can be used as a guidance to follow a set of user profiles. After data pre-processing, a knowledge base is set up by mining association rules from the database which will then be used along with the travel request to assign a score to each of the potential travel offers, thus ranking them. The research is performed as a part of the Ride2Rail project in the frame of the Shift2Rail initiative as part of the Innovation Programme 4 of EU Horizon 2020.

Keywords Context .Preferences .Travel .Data Mining .Recommender Systems.

Sommario

La personalizzazione dell'esperienza dell'utente attraverso raccomandazioni implica la comprensione delle sue preferenze e del contesto in cui queste preferenze sono presenti. In questo lavoro, miriamo a progettare un sistema che, a fronte di una richiesta fatta da un possibile cliente, presenta una serie di offerte di viaggio opportunamente classificate. È importante conoscere le preferenze nel tempo e utilizzare queste informazioni per comprendere le esigenze del viaggiatore. La nostra soluzione si basa su un sistema di raccomandazione basato sul data mining che imposta una base di conoscenza che verrà utilizzata insieme alla richiesta di viaggio per assegnare un punteggio a ciascuna delle potenziali offerte di viaggio, classificandole in tal modo. La ricerca viene svolta nell'ambito del progetto Ride2Rail nell'ambito dell'iniziativa Shift2Rail nell'ambito del Programma per l'innovazione 4 di EU Horizon 2020.

Chapter 1

Introduction

This chapter provides an overview of the thesis. In Section 1.1, we explain the context and motivations which led us to conduct this research. In Sections 1.2 and 1.3 we specify the objectives and summarize the proposed solution to address them. Then, Section 1.4 presents information about the structure of the thesis.

1.1 Context and Motivations

With the abundance of data that is available from various sources, personalization of applications and services has become the need of the hour [14]. Recommender Systems, as a means of personalisation, offer suggestive mechanism to keep the user engaged [4] and at the same time retrieve information from them based on their activities. Information required to personalise an experience could be static profile data as well as evolving preferences of the user. Moreover, the environment and conditions around the user play a role in filtering the information presented to the user. The level of engagement and personalization comes with time when the system knows the user completely and can predict his/her choices [51].

1.2 Objectives

This research is performed as a part of the Ride2Rail project in the frame of the Shift2Rail initiative as part of the Innovation Programme 4 of EU Horizon 2020. The objective of our work is to be able to provide a personalised set of travel offers to a traveller, based on the context they are present in and their preferences. The aim is to learn the preferences over time and use this information to understand the traveller and present the offers ranked suitable to their needs.

1.3 Proposed Solution

The solution is to implement a data mining based recommender system. We start by designing the features of the application through a database design of historical traveller data including travel offers. The database is then populated, while guiding them to follow some inbuilt characteristics by fixing possible values and applying constraints. After pre-processing the data, a knowledge base is set up by mining association rules from the database. The rules are designed to address new users with no past preferences or past travel history. The knowledge base will then be used along with the travel request to compare and assign a score to each of the potential travel offers. The overall calculated score is used to rank the offers and provide it to the traveller.The knowledge base of association rules works to mine the user preferences and understand the traveller's needs.

1.4 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 explains some theoretical background regarding the applied methods in the thesis, Chapter 3 provides state of the art methodologies working with context aware systems. Then, in Chapter 4, we discuss the steps to build a data mining based ranking module. Chapter 5 reports and explains the obtained results. Finally, in Chapter 6 we provide the conclusion and future works.

Chapter 2

Background

In this chapter, we provide the reader with the basic theoretical framework and concepts used in this thesis.

2.1 Travel Companion

This module is being developed as a part of the Shift2Rail and the EU Horizon 2020 research and innovation programme. It helps the user by assisting in all phases of his/her travels, from planning, to execution, to post-trip operations. In addition, TC stores all information (travel choices, tickets, etc.) related to the trips organised and taken by the user. It also works with retailers and operators who will be able to identify and authorise the TC to access their own systems and networks. TC provides a personalised experience for the traveller based on context and their preferences.

Context is a set of parameters that we use to filter the travel offer data that we provide to the traveller. It can be identified as a set of features contributing to the decisions of a user in a system [6]. Example: location, time, weather etc. Preferences is a set of parameters that we use to rank the travel offer data. Example: Meal preference, Service preference etc. These could change over time or remain static depending on the type of preferences. For example, a health issue could be specified for a period of time while a meal preference could change over time.

A Travel offer is a ticket indicating Source, Destination, mode of transportation and also includes descriptive information (feature/variables/characteristics) regarding CO2 emission, price, and many other features which provide information about the facilities it provides as a part of the offer. For example, meal options available, allowed luggage etc. These correspond to the preferences mentioned by the user and hence play a role in being used to rank the offers. In regard to the traveller, there is static profile data, personal preferences, and travel requests made. A Travel Request includes Source, Destination, purpose for travel and some additional information about the request like trip frequency. The information put together allows the system to further understand the traveller needs and general behavior over time.

2.2 Recommender Systems

Recommendations are an important part of providing the personalised experience and several techniques can be used. These differ on how the information about the user is used to provide recommendations. In particular, we can use *content-based filtering* [39], *collaborative filtering* [28] and *hybrid filtering* [13].

Content-based filtering systems provide suggestions to the user based on his own preferences. It further understands the user based on his past choices made based on the suggestions. In contrast, collaborative filtering systems base their recommendation for a given user on the past preferences of other users who share similar preferences. This technique relies on public opinion and works on correlating groups of users based on preferences. For the former, accuracy depends on a training period in order to detect user preferences. Hence, it needs prolonged use of the system by the user to learn his/her preferences. For the latter, accuracy depends on the availability of a huge data set from a diverse set of users to be able to include all user types. Hybrid techniques, which combine content-based and collaborative filtering leverages the benefits of both techniques and is more commonly used.

Another approach to recommendation is *data mining based recommendation* [1]. Instead of training a model to understand the user, this approach learns behavioural rules from stored past transactions and and recommendation is based on the learned rules acting as a knowledge base [4].

RS are designed to suggest options to users considering their requirements. The *user profiling* approaches, that emerged in the literature with the aim to determine the users' needs and activity patterns [35], fall into one of the following categories: *Explicit* approaches, often referred to as *static user profiling*, predict the user preferences through data mostly directly obtained from them; *Implicit* approaches, instead, mostly disregard the users' static information and rely on the information obtained from observing their behaviors indirectly; *Hybrid* approaches are a combination of the two [40]. Generally, *Explicit* approaches are hard to follow as most users do not like to fill up forms or surveys and the information obtained might be incomplete. It is advised to follow a *Hybrid* approach to maximise the data obtained.

2.3 Context-awareness

Context playing a vital role in personalisation, introduces the need for incorporating context into the information system. Various models for planning context-aware systems have been described in many surveys [7, 30]. Each provides interesting insights into including contextual information based on application requirements.

The work [8] introduced the Context Dimension Tree (CDT) model and a related methodology aimed at representing, and later leveraging, the information usage of *contexts*, in order to cover different situations in which the user can act, and picturize them hierarchically as a rooted labeled tree. The CDT model has the flexibility to capture both in the *conceptual* and *detail* level. An example of a CDT is depicted in Figure 2.1; [32] has proposed this; the root of the tree represents the most general context, N is the set of nodes, which are either black *dimension nodes* N_D or white *concept nodes* N_C *a.k.a* dimension's values; N_D and N_C must alternate along the branches. White squares are *parameters*, shorthands to represent the nodes that have many possible values. Dashed lines specify if the N_C children of N_D are mutually exclusive. The children of r define the main analysis dimensions and are known as *top dimensions*. Each N_D should have at least one N_C. This model offers to completely capture the context information for the system and also includes the possibility of checking the relevance between contexts by following the hierarchy. Any further interesting features can be added by increasing or decreasing the level of granularity of the model.

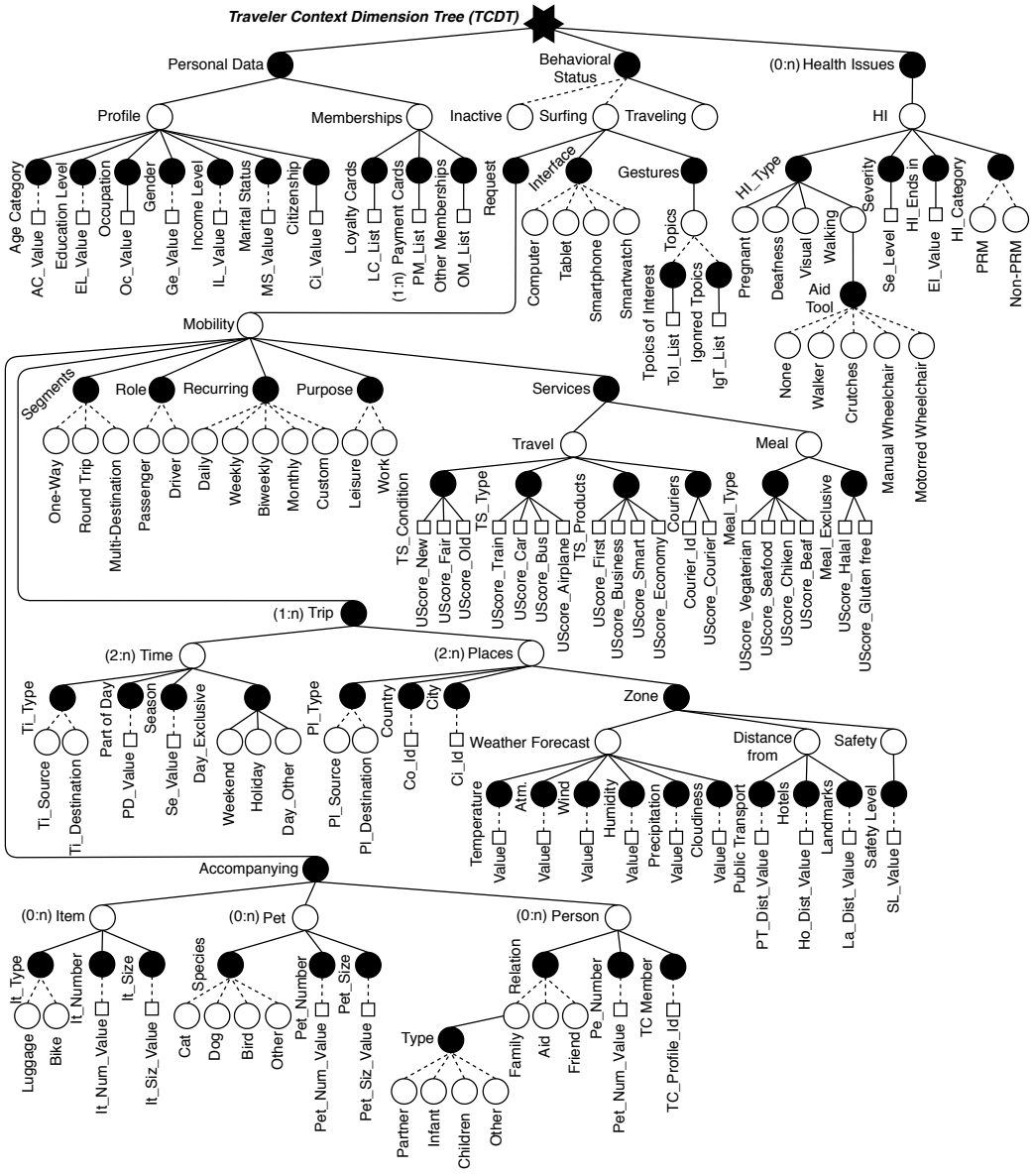

Figure 2.1: Traveler Context Dimension Tree [32].

2.4 Preference Learning

Personalisation of user experience through recommendation involves understanding their preferences and learning them over time. The techniques previously discussed in Section 2.2, *Explicit* approaches and *Implicit* approaches are about fetching the preferences from the user whereas *content-based filtering* and *collaborative filtering* are in reference to learning them.

The learnt preferences can be stored as vectors or rules which can be retrieved in relevance to the user, with context and then used to score and rank the recommendations to be provided to the user. The score represents how well the recommendations match and thus the ranking presents an ordered set of suggestions to the user. The work [3] shares some approaches to preference learning in a dating application where profiles have been matched solely based on preference learning.

2.5 Frequent Itemset Mining

Data mining is the process of fetching hidden information found in data and therefore it can be seen as a crucial step in the knowledge discovery process. It includes different functions like clustering, classification, prediction and link analysis.

Frequent pattern mining is the process of mining data in a set of items or some patterns from big databases, which must cross the minimum support threshold (The number that filters based on the frequency of the data). A frequent pattern is a pattern that occurs recurrently in a dataset.

Frequent itemset indicates that a set of items that frequently occurs together in a transactional data set. For example, bread and butter in a retail store transaction.

Frequent pattern mining was first proposed by [2] for market basket analysis in the form of association rule mining. It analyses customer purchasing behavior by finding associations between the different items that customers bought.

Let I =$\{ i_1, i_2, ..., i_n \}$ be a set of all items. A k-itemset α, which consists of k items from I, is frequent if α occurs in a transaction database D no lower

than $\theta|D|$ times, where θ is a user-specified minimum support threshold, and $|D|$ is the total number of transactions in D.

Frequent itemset mining plays an important role in many data mining arenas such as association rules, warehousing, correlations, clustering of high-dimensional data, and classification [47].

2.6 Association Rule Mining

Association rule mining is a rule-based data mining technique for discovering interesting associations between items in large databases. The method uses frequent itemsets as a base to discover the rules. These rules are in the form of if-then statements that gives information about the probability of relationships between items. For example, in market-basket anlaysis, it is useful to understand that if a customer has bought bread he/she is likely to buy butter.

Initially introduced in [2], given a set of items and a set of transactions that each contains a subset of the items, an association rule is defined as an implication of the form $A \implies B$ where A, B are the subsets and $A \cap B = 0$. The sets of items (for short *itemsets*) A and B are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule.

Association rule mining is the process of finding rules which fulfil the predefined minimum support and confidence from a known data base. According to [47], Support corresponds to the frequency of the set $A \cup B$ in the dataset and confidence corresponds to the conditional probability of finding B having found A, and is given by $(supA \cup B) \| sup(A)$. The item set which supports the minimum support and confidence is known as frequent itemset.

The process of mining the association rules can be divided into two fragments: First, Invention of all frequent item sets. Second, Generation of strong association rules from the frequent item sets. Some of the popular algorithms for association rule mining are Apriori, Eclat and FP-Growth. They differ in the first step of mining frequent itemsets.

2.7 Apriori algorithm

Apriori is one of the most popular algorithms for generating frequent itemsets from the dataset [52]. Association rules are then mined based on the frequent itemsets with confidence greater than or equal to a minimum threshold. The rules forms the knowledge base that would be leveraged by the recommendation system.

The database used as input should follow a horizontal orientation of providing each transaction with the associated items in lexicographical order. The algorithm is based on the concept of candidate generation with a join based approach.

Let the set of frequent itemsets of size k be F_k and their candidates be C_k. Apriori first scans the database and searches for frequent itemsets of size 1 by calculating the frequency for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets.

1. Generate C_{k+1}, candidates of frequent itemsets of size k+1, from the frequent itemsets of size k.
2. Scan the database and calculate the support of each candidate of frequent itemsets.
3. Add those itemsets that satisfies the minimum support requirement to F_{k+1}.

The algorithm stops when there are no more combinations left. It runs at most k_{max+1} times when the maximum size of frequent itemsets is set at k_{max}. The reduction of size of candidate sets helps in achieving good performance. It is quite a simple algorithm and easily implemented in many programming languages.

2.8 Dataset

For a solution based on data mining, the important requirement is a huge database with collection of historical data from users. These serve as the transactional database that can be used to mine the association rules for the knowledge base.

With regard to our system, we need data that includes information about the traveller's profile, past preferences, travel requests, potential travel offers and their characteristics along with the final choice that was made with each request. The diversity of the travellers and past trip information ensures wide coverage of the rules to be able to make accurate suggestions for a new traveller. This dependency on historical data is referred to as *cold start problem*.

Possible solutions to this problem are to use publicly available datasets or synthesising/generating a suitable dataset. The former might have compatibilty issues with the system to be solved while the latter could provide opportunities to train the data in order to be able to validate the algorithms and methodology in use.

Publicly available datasets are more often than not, survey information obtained from users filling forms. This data might be incomplete leading to data scarcity. Moreover, the attributes that were in focus during the data collection are already fixed and might not sync with the system under consideration. Synthesising datasets is done by programmatically generating data thus controlling their distribution and randomness that is necessary for a robust database.

Chapter 3

Related Works

In this chapter we provide some of the relevant state of the art methodologies and techniques for context aware recommender systems.

3.1 Context Models

Representation and visualisation of context information sets up the stage for a context aware system. *A Data-oriented Survey of Context Models* [7] provides an insight into comparing various models in reference to a target application and *data tailoring* has been used as a sample application. The comparison is based on aspects of context dimensions, representation features and context management.

The context dimensions that are taken into account are space, time, context history, subject and user profile. The representation features includes some general characteristics of the model like type of formalism, flexibility, level of formality, variable context granularity and valid context constraints. Context construction, reasoning, information quality, ambiguity, multi-context modeling and automatic learning features explains the way the context is built, managed and exploited.

The systems compared with reference to data tailoring are ACTIVITY [34], CASS [22], CoBrA [16], CoDaMoS [41], COMANTO [45,50], Context-ADDICT [5], Conceptual-CM [17], CSCP [12], EXPDOC [48], FAWIS [20,21], Graphical-

CM [27], HIPS/HyperAudio [49], MAIS [7, 38], SCOPES [37], SOCAM [24] and U-Learn [53]. The analysis provides all research necessary for a designer to choose among the discussed models or in order to set up a new context model.

3.2 Context-aware View

CARVE- Context-aware automatic view definition over relational databases explains a methodology to define the contexts followed by associating them with the relevant information. It aims to facilitate applications requiring to access different parts of the database based on their current situation [9].

The idea helps the application code in not changing anything at database access but designs context-aware views over the global schema that adapts in sync to the environment changes. CARVE uses the CDT model for the context design phase with the relational database and keeps query resilience and set orientation in check during the automatic view combination. It explains the process in detail using a demonstration with a food delivery application and also includes a prototype tool called *CADDFrame* to support the data tailoring at design phase.

3.3 Context-aware Access to Heterogeneous Resources

Latest data integration systems put together data fetched from different kind of sources and display it through all kinds of devices. To bring in filtering based on context, [18] presents a design framework that helps in context specification and mapping them into on the fly data integration that can be embedded into applications. It explains the methodology in detail using a demonstration with a tourism app for a traveller.

The data integration involves combining data from internal data sources, external services and mapping onto a resource schema along with CDT as the context model. Since the queries and resource will be *context agnostic* and CDT is responsible for the correspondence, the queries have to be rewritten. The paper also incorporates a *node.js* prototype that consists of a context

manager, query manager and response aggregator and their performance measures for service selection and invocation.

3.4 Context-aware Data Design

Different context models have been discussed and classified based on usage [7]. Context as a matter of channel-device-presentation, location and environment, user activity, sharing and selecting relevant data, functionalities and services. Work in [8] talks about context being considered as a multidimensional space that works together to retrieve the context relevant data.

It introduces the concept of context dimension tree and further explains on adding constraints, *forbid constraints* and *granularity-level constraints* that helps in enriching the combination of useful context dimensions for a given scenario. The tree model CDT is completely independent of representation, which could be XML or ontology etc. and can represent with many levels of abstraction.

3.5 Context-based Personalisation

Information systems in organizations are continuously changing with time and with introduction of newer technologies to represent data. This results in overload of information and it is the need of the hour to be able to tailor this according to the user and remove all irrelevant data. Context becomes the vital tool for personalising and providing precise information. *context-guided methodology* helps the designer of the information system to be able to categorise and associate data based on context.

The two techniques discussed are called *configuration-based mapping* and *value-based mapping* [6]. They are distinguished based on the direction that was chosen for data view production. The former is more manual although precise while the latter is automatic but could be prone to errors. The suggested methodology could also be extended to design larger information systems like Data Warehouses.

3.6 PREMINE

Premine [36] that is a JAVA-based tool, uses a relational database (movies) and provides a data mining approach to learn contextual preferences of users based on the movies they preferred and the movie characteristics they were interested in. These learnt preferences are then used to personalise the set of data associated with each context.

Context model used is CDT and preference model is categorized as σ *preferences* and π *preferences*. The former refers to preferences on tuples (movies) and the latter refers to preferences on attributes (characteristics). Contextual preference is a pair (C,p), where C is a context and p is either a σ *preference* or π *preference*.

Server log, a set of tables to maintain the transactional data is stored, separately for σ *preferences* and π *preferences* and is synchronised periodically. Association Rules are mined from the server log using *Conqueror+* and $FP-growth$ algorithm. Each of these rules are assigned a score based on their frequency and relevance to context. The rules along with their score help in creating the knowledge repository that is later used in the contextual view personalization process. The usage of CDT and association rule mining for knowledge base provides us insight for our data mining based recommender system.

3.7 Case Study by OMIO

Omio[1] ranked airports in Europe that offer a relaxing and comfortable travel experience. The experiment was conducted by examining all Eurpean capital cities. The analysis was based on 15 factors that was clustered into *connectivity*, *facilities*, *entertainment* and *stress*.

The result of each of the factors are normalised to score from $0-100$. The clusters that were determined by airport authorities like facilities and entertainment, weighed 40% each of the final score while stress and connectivity weighed 10%. All the results were finally accumulated by country and then standardised on a scale from $0-100$. Clustering based on a given set of

[1]https://cdn−goeuro.com/static_content/web/content/User_Friendly_airports/UK-Methodology%20_Most%20Convenient%20Airport_%20from%20Omio.pdf

factors is a widely popular technique in data profiling and could also be extended to user profiling based on context and preferences, with scoring to reflect their relevance.

3.8 RARE

RARE [4] is a course **R**ecommender system based on **A**ssociation **R**ul**E**s, which involves a data mining process improved with user ratings as a feedback mechanism. With a database of historical student data, rules are mined to link courses that were taken up by senior students as suggestions for the new/junior students.

In order to improvise this RS, ratings are received from students to understand their opinion. The ratings are incorporated into an evaluation mechanism that helps in removing recommendations which are not well received. Feedback is also fetched by calculating *accuracy* and *coverage* from the recommended courses and courses followed by each student. These two factors are used to measure the performance of the system and a compromise is reached between the two factors to maximise performance.

RARE overcomes the cold start problem by setting up basic knowledge of extracted rules from borrowed real anonymous data from a university. It allows scalability by keeping the mining process off-line. The rules base could further be compacted to save memory by tweaking the minimum support and confidence.

3.9 User Profiling

User Profiling is a technique of recommender systems where it learns about the user and suggestions are then based on the learnt user information [35]. It can be broadly categorised into *Explicit user profiling, Implicit user profiling* and *Hybrid*. Explicit is when the user provides his own information voluntarily through forms/surveys, Implicit is when the system indirectly tries to capture information about the user through observations and Hybrid is a combination of the two.

The steps involved in user profiling are *profile extraction* where user information is fetched from different sources, *profile integration* is processing of

the fetched information and *user interest discovery* finally categorises the users into clusters based on their characteristics. The process of categorisation is called filtering which can be based on the user's personal preferences *(content based filtering)* or on other user's preferences who share a similar interest *(collaborative filtering)*. User profiling has a wide range of applications from recommender systems in tourism, search personalisation and energy management.

3.10 Context Awareness in Travel Companion

In Section 2.1 we talked about Travel Companion in brief. The work [33] presents elements of a recommender system for the TC module. The paper mentions the proposed Context Dimension Tree for the same and a conceptual overview of the system architecture. Figure 3.1 shows the proposed idea envisioned with all main actors of the system.

The main parts, that are of close interest to our work in consideration are the Recommender Core and Data block. They include the knowledge models, preference learner and evaluation metrics that we will be implementing in our system. The paper suggests on how trip recommendations are provided to a traveller based on the information available regarding the user preferences. It considers a user priority by fetching a score for each of the preference options available under the CDT dimensions. This are used as personal constraints to filter some options from the list of travel offers and then rank the remaining based on some evaluation metrics.

The TC incorporates the S2R *Interoperability Framework* of the Sprint Project [46] and extension of domain ontology to facilitate data sharing between TC and other modules through ontology-based annotations and mappings [15, 31]. It includes a Social Media Core that facilitates fetching on travel related information from social media while being able to use it for marketing campaigns. The external services that will be in connection are shown in the *Third Parties* block. The information regarding the connection that the blocks have with each other helps in understanding the overall functionality of the system.

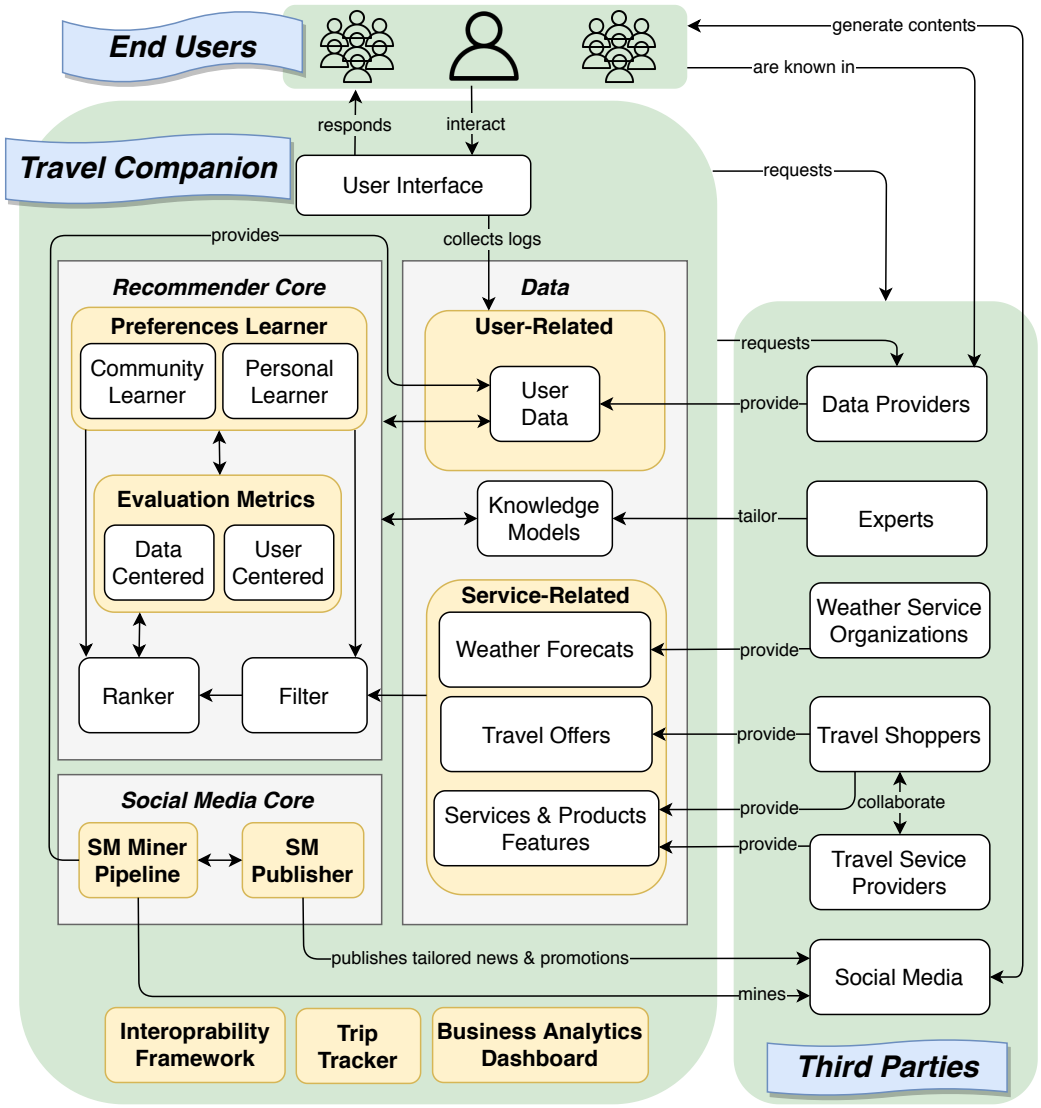

Figure 3.1: Proposed Conceptual Architecture [33].

3.11 Social Media Mining

Data obtained from social media in the field of travel could gave great insights into real traveller information and their characteristics. It is a form of collecting public opinion, current travel trends and helps in modelling traveller behaviour. Work in [44] mines such information available from various sources like *facebook, instagram, twitter* etc.

The travel attributes fetched like age, gender, mode of transport, trip purpose etc. are in sync with our work and is helpful in leveraging social media to populate our database. This would be a cost effective solution compared to organising a survey by ourselves suitable to our application requirements. While there can be issues of data privacy, it can be reduced by cautiously collecting overall behaviour and being anonymous to individual information.

Complimenting the usage of social media for information retrieval, it includes positive results of a survey from travel demand modelling experts from different parts of the world, on usage of social media data for formulating daily travel behaviour.

3.12 Recommendation and Ranking

Recommender Systems in the travel industry helps to cope with the demand for touristic services and information by online. This work [51] implements such a system using collaborative, hybrid recommendation approach on real travel data. The work discusses the Tourist-Area-Season-Topic (TAST) model and extends it by adding the relationship between tourist groups. It divides the tourists into groups by following *K means clustering algorithm* and provides targeted recommendation.

Another application of Hybrid Group Recommender System is done by [19] where they provide destination suggestions based on user preferences and past ratings as feedback. It uses the freely available data on travel, *WikiVoyage* [2] as its main information source supported by popularity rating from *Tripadvisor*,[3] an American travel website providing reviews of travel-related content. Group recommendation [23] combines profiles from a family or a group of friends to leverage all their information specifications. Other interesting works like [29] uses a recommendation algorithm to combine multiple travel destinations into one trip while taking into consideration, budget and time constraints.

[2]http://dumps.wikimedia.org/enwikivoyage
[3]http://www.tripadvisor.com

Chapter 4

Methodology

In this chapter, we will introduce the system and how it was implemented. A global structure of the process is provided, along with the detailed explanation of each step. All the tools, the coding language and the relevant libraries used for developing the system will also be listed.

4.1 System Overview

Our goal for the system is to be able to rank the travel offers such that it is ranked personally for the traveller based on the context and their preferences. The recommendations for a new traveller will be based on the historical data of other travellers with similar profile and preferences. The system also aims to learn the preferences over time. Hence, the solution is to build a recommender system that sets a knowledge base which will be used along with the travel request to assign a score to each of the potential travel offers. The score is then used to rank the offers and provide it to the traveller. Figure 4.1 shows the overview.

The chosen recommender system for the solution is *data mining based recommendation*. Hence, we need to mine association rules from a huge database with collection of historical data from travellers as mentioned in Section 2.8. In order to provide a warm start for new users we need an existing dataset. Our first approach was to find a suitable publicly available dataset. Although the datasets available had some information about travellers in Europe and

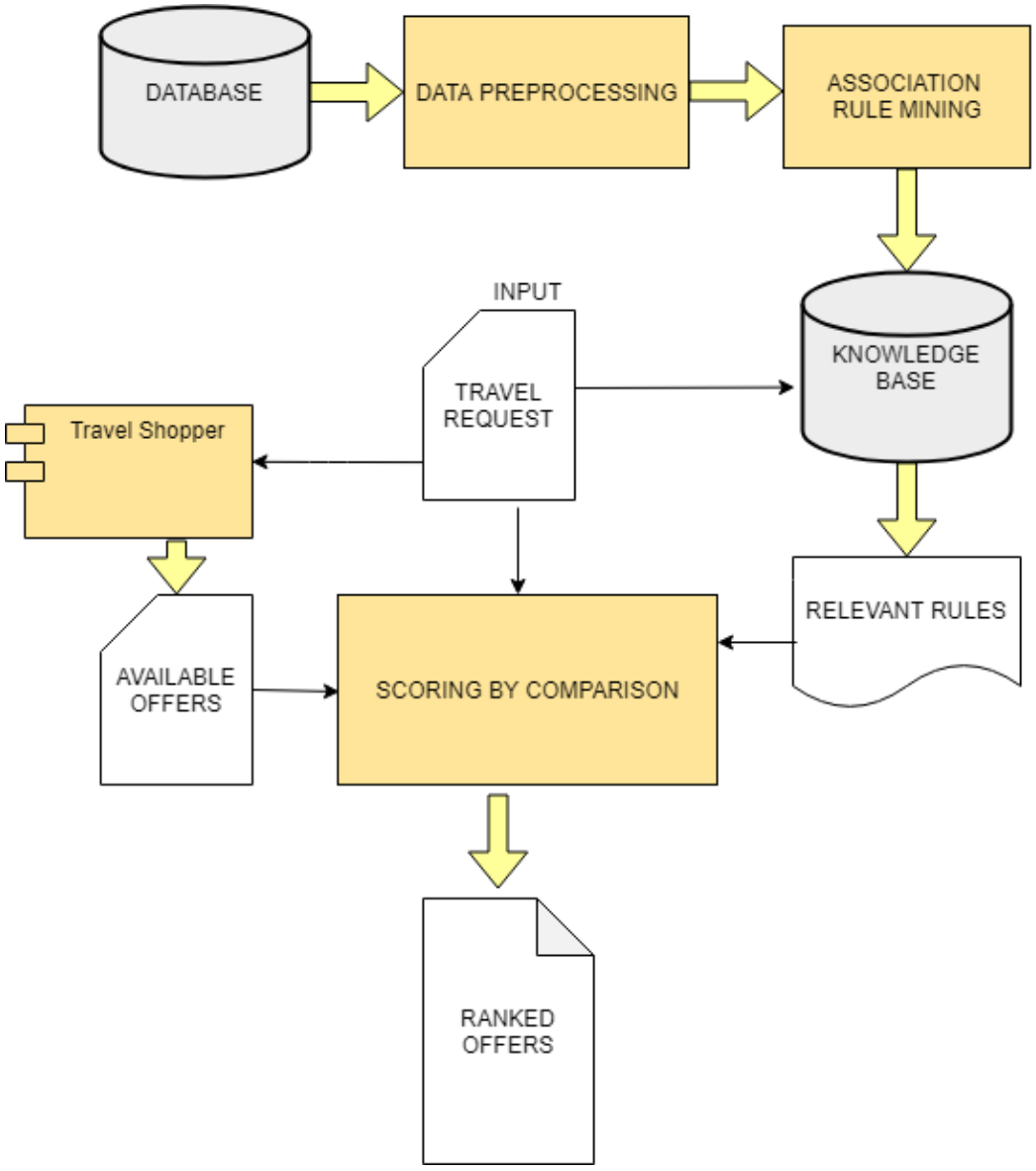

Figure 4.1: System Overview Diagram.

public transport facilities, we encountered some compatibility issues with our system. We then decided to synthesis our own dataset and the resulting database can be further extended as the system backend once the travellers are acquainted with the system. Also knowing the distribution of the dataset allows us to validate the results.

The system implementation consists of:

1. **Database Design**
2. **Data Population**
3. **Knowledge Base**
4. **Ranking**

Figure 4.2: Process flow of the system implementation.

Figure 4.2 shows the process flow. Each of the steps are explained in detail in the upcoming sections.

4.2 Database Design

The database should include information about the traveller's profile, past preferences, travel requests, potential travel offers and their characteristics along with the final choice that was made with each request.

- Traveller's profile information consists of name, age, mobile number, gender, education, occupation, marital status, card details (membership/debit/credit) and citizenship.
- Preference information consists of health issues, accompanying partner information, service preferences, meal preferences and luggage requirements.

- Travel request information consists of purpose of travel, source, destination, role as a passenger/driver, frequency of the trip if recurring and the interface used to make the request.
- Travel offer information consists of an identifier to map it to the request, source, destination, number of segments, duration, price and discount. It also includes information about pets being allowed in the journey.
- Offer characteristics includes the special facilities it includes, service type, meal availability and luggage facility.
- Each request is also linked to the final choice of offer that was made by the traveller.

All information gathered and explained are then divided into entities and relationships. They are presented as follows: Entities and Relationships are highlighted in bold with their attributes highlighted in italic.

ENTITIES

1. **Traveller**- *Mobile number, Name, Age, Education, Gender, Occupation, Marital status, Citizenship*
2. **Card** - *Card number, Card type, Username, Expiry date*
3. **Health Issues** - *Issue type, Severity, Category, Aid*
4. **Request** - *Purpose, Role, Leg, Recurring, Source, Destination, Interface*
5. **Accompanying** - *Category, Type, Number*
6. **Service** - *Type, Condition, Class*
7. **Meal** - *Meal type, Excluded*
8. **Luggage** - *Luggage type, Number*
9. **Travel Offer** - *Offer ID, Tag ID, Source, Destination, Duration, Number of segments, Pets allowed, Price, Discount*
10. **Special facilities** - *Type, Detail*

RELATIONSHIPS

1. **User health** - *connects Traveller and Health Issues with time duration*
2. **User partner** - *connects Accompanying and Traveller with timestamp*
3. **Membership** - *connects Card and Traveller*
4. **Service preferences** - *connects Service and Traveller with timestamp*
5. **Meal preferences** - *connects Meal and Traveller with timestamp*
6. **Luggage preferences** - *connects Luggage and Traveller with timestamp*
7. **Travel request** - *connects Request, Card, Travel Offer and Traveller with timestamp*
8. **Offer detail** - *connects Travel Offer, Service, Meal, Luggage and Special facilities*

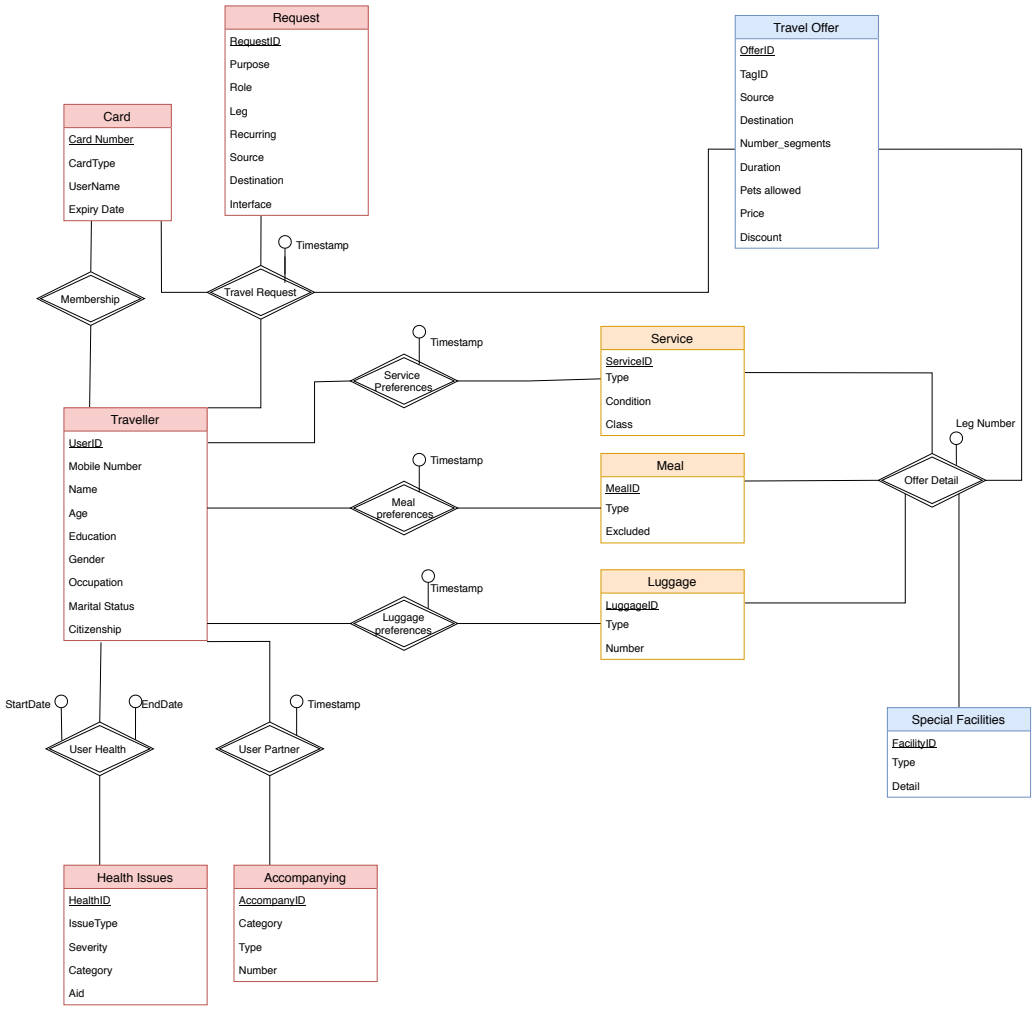

Figure 4.3: Entity Relationship Diagram of the Database Schema

Figure 4.3 shows the Entity Relationship Diagram of the discussed Database design. It shows the structure of our database and gives an insight into what the entities hold (attributes) and how they are connected by relationships. The entities are represented by a rectangle and relationships are represented by a diamond shape. The underlined attributes denote the primary key of the Entity. The ER Diagram is then converted to Relational Schema.

Figure 4.4 shows the Relational schema of the discussed Database design. It represents data as a set of related tables. The entities and relationships are both written as tables. The primary keys of the connecting entities form the foreign key of the relationship tables along with other attributes of the relationship if necessary. The conversion from ER Schema to Relational Schema helps in the next step of implementing the database. The tables, attributes, primary keys and foreign keys are all understood from the relational schema. It also provides an idea of the size of the database which plays a major role in deciding the technology for the database.

TRAVELLER(UserID,MobileNumber,Name,Age,Education,Gender,Occupation,MaritalStatus,Citizenship)
CARD(CardNumber,CardType,Username,ExpiryDate,UserID)
REQUEST(RequestID,Purpose,Role,Leg,Recurring,Source,Destination,Interface)
USER_TRAVEL_REQUEST(UserID,RequestID,Timestamp,OfferID,CardNumber)
SERVICE_PREFERENCES(UserID,ServiceID,Timestamp)
MEAL_PREFERENCES(UserID,MealID,TimeStamp)
LUGGAGE_PREFERENCES(UserID,LuggageID,Timestamp)
HEALTH_ISSUES(HealthID,IssueType,Severity,Category,Aid)
USER_HEALTH(UserID,HealthID,StartDate,EndDate)
ACCOMPANYING(AccompanyID,Category,Type,Number)
USER_PARTNER(UserID,AccompanyID,Timestamp)

SERVICE(ServiceID,Type,Condition,Class)
MEAL(MealID,Mealtype,Excluded)
LUGGAGE(LuggageID,LuggageType,Number)

TRAVEL_OFFER(OfferID,TagID,Source,Destination,Duration, Number_segments,Pets allowed,Price,Discount)
SPECIAL_FACILITIES(FacilityID,Type,Detail)
OFFER_DETAIL(OfferID,Legnum,ServiceID,MealID,LuggageID,FacilityID)

Figure 4.4: Relational Schema of the Database Design

User health, is a *many-many* relationship and needs a *startdate* and *enddate* to let the system know about the duration of the health issue. *User partner, Service preferences, Meal preferences and Luggage preferences* are also *many-many* relationships and they need a *timestamp* attribute to let the system know when the preferences were added so that the most recent ones can take priority. *Membership* is 1-1 relationship and can be linked by adding the userID in *Card* table. *Offer Detail* is definitely a *many-many* relationship connecting all the facilities, an offer can provide along with a leg number to denote the segment of the journey, in case of multi modal transport. *Travel request* is again a *many-many* relationship with a *timestamp* to denote the time of request and connects to offer to let the system know which offer was chosen as the final choice.

4.3 Data Population

In order to populate the tables as per the database design discussed previously, the strategy followed was to split the process in 2 steps.

1. Populate independent tables
2. Populate relationship tables

Explaining these steps further in detail, Section 4.3.1 talks about populating independent tables, Section 4.3.2 talks about populating relationship tables, Section 4.3.3 talks about the database technology chosen for storage along with the programming language used for the data population and Section 4.3.4 mentions some characteristics about the populated dataset.

4.3.1 Populate Independent Tables

Independent tables include Traveller, Card, Health Issues, Accompanying, Request, Service, Meal, Luggage, Travel Offer and Special Facilities. We need some rules in place to avoid invalid data. For example, a male traveller with a pregnancy health condition. Moreover, the data should be directed/-trained in following a direction and some inbuilt characteristics that could be validated later with the results.

The possible values that the tables can hold are as follows.

Traveller
Gender - 50% M/F
Age - (18-100)
Citizenship - Covering EU
Education - midschool/highschool/graduate/postgraduate
Occupation - student/worker/professional/business
Marital status - single/married

Card details can be added for every traveller data

Constraints to be added to remove invalid data that could come up from random combination of age ,education, occupation.

Request
Purpose - work/leisure
Role - passenger [1]
Leg - one way [2]
Recurring - daily/weekly/monthly
Source and Destination - covering cities in EU
Interface - smartphone/tablet/PC

Travel offer

5 offers to be generated for each request.
Source Destination - from *Request*
Pets allowed - yes/no
Discount - card types (premium/student/frequentflyer/europass)
No of segments (1-3) $\implies$ 0 to 2 stops

Service
Type - Train/Car/Bus/Airplane
Condition - New/Fair/Old
Class - First/Business/Smart/Economy

Meal
Type - Vegetarian/Chicken/Seafood/Beef
Excluded - Halal/Gluten Free

[1] role as driver implies blabla car and that would be considered in future work to include putting up requests for passengers while initiating a trip

[2] leg as round trip and multi city will hinder source destination values, complicating it, which would be another future work extension

Accompanying
Category - Pet/Person
Type - cat/dog/bird(pet) family/aid/friend
Number - (1-2)

Luggage
Type - Bag/Bike
Number - (1 - 3)

Special facilities
Type - Visual/Walking/Deafness
Detail - Brail/wheelchair dock/text signal

Health Issues
Type - Pregnant/Visual/Deafness/Walking
Severity - low/medium/high
Category - PRM/Non PRM
Aid - Walker/Crutches/Wheelchair (N/A for pregnant/deafness/visual)

4.3.2 Populate Relationship Tables

Relationships tables include User health, User partner, Membership, Service preferences, Meal preferences, Luggage preferences, Travel request and Offer detail

The possible values that the tables can hold are as follows.

Travel Request - Different types of request should be generated for each traveller differing either by purpose or source/destination values.

User Health - Each health issue can be added in a single severity level with a single aid. Male user with pregnancy health condition cannot be possible.

User Partner - The "aid" type can be added only with the presence of health issue else the partner should be family/friend/pet.

Service, Meal, Luggage Preference - To cover all types of preferences for each traveller and avoid repetitions.

Offer Detail - For each offer, and for each of its segment, service type, meal availability, luggage facility and special facility details should be added.

They are grouped together into one table as they are tied together with the offer.

4.3.3 Choice of Database and Language

Once the database design was ready along with the rules to populate the data, the next step was to decide the type of database and the programming language to be used.

PostgreSQL [3] is a powerful, open source object-relational database system that uses and extends the SQL language. It can robustly store and scale huge data workloads. We chose PostgreSQL as it runs on all major operating systems and has been ACID-compliant since 2001. It is fault tolerant and is the popular choice of developers to build applications as well as administrators to protect data integrity. Postgres is also highly extensible and allows definition of custom datatypes, functions and different programming languages.

Data population module was implemented in **Python** which is a high-level open-source programming language, popular for its simplicity. It has its usage in development as well as scripting and works on different platforms (windows, MAC, Linux etc). Python started with version 2.0 and the latest version is 2.7. However, Python 3.0 was a separate release with its newest version 3.6 used for this work. Being open source, it has immense support from the community and almost every function requirement has a potential library already implemented.

For our solution, the following modules were used.

psycopg2 is the PostgreSQL database adapter for the Python programming language. It completely implements Python DB API 2.0 specification along with thread safety. psycopg2 is mostly implemented in C as a libpq wrapper, resulting in being both efficient and secure. We used it in our system to generate data using Python and insert it into our PostgreSQL database.

requests sends HTTP requests easily. There is no manually adding query strings or encoding POST data, everything is in JSON format which is easy to use. We used it to get information on cities and countries in EU.

[3] https://www.postgresql.org/

random simply implements pseudo-random number generators for various distributions including integer and float. We used it to choose options for preferences, offer facilities and age.

faker is a Python package that generates fake data. **faker.providers.profile** generates a complete profile. **faker.providers.date_time** generates a date time object with multiple variations like from the past, future, month, days, year etc.

Constraints were used to keep the rules in place while generating random combinations of data.

```
DELETE from request WHERE source like destination;

UPDATE traveller SET education='highschool' WHERE age
   <21 AND education like 'postgraduate';

UPDATE traveller SET education='highschool' WHERE age
   <21 AND education like 'graduate';

UPDATE traveller SET education='graduate' WHERE age<26
   AND education like 'postgraduate';

UPDATE traveller SET occupation = 'worker' WHERE
   education like 'midschool' AND occupation like '
   professional';

UPDATE  traveller SET occupation='professional' WHERE
   education like 'postgraduate' AND occupation like '
   worker';

UPDATE traveller SET occupation='business' WHERE age>60
   ANDd occupation like 'student';

UPDATE traveller SET occupation='professional' WHERE
   age>50 AND occupation='student';
```

4.3.4 Dataset Characteristics

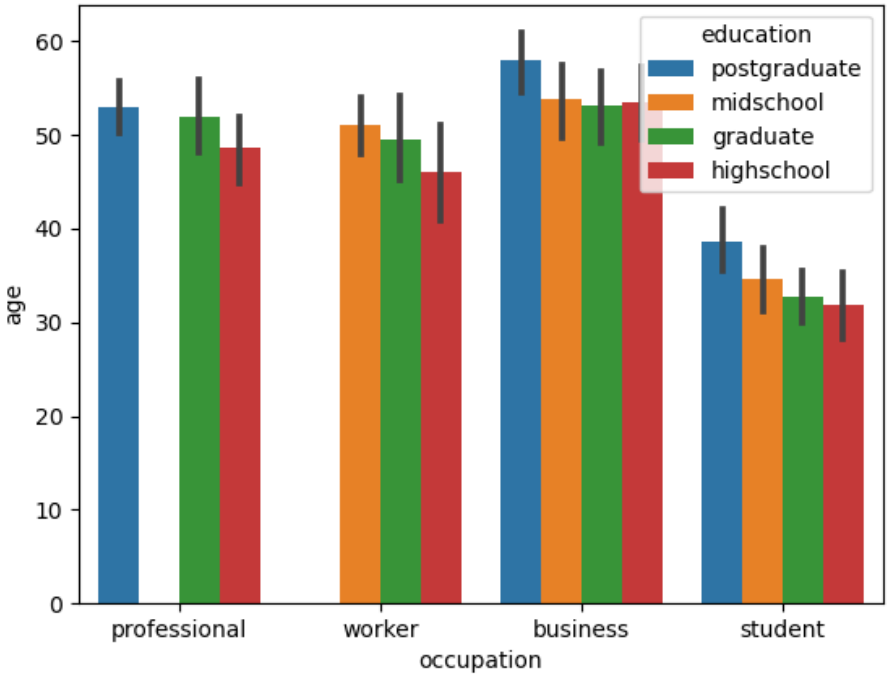

Figure 4.5: User profile data distribution

The populated dataset was a collection of data based on the combination of $1,000$ traveller profiles, $2,000$ travel requests and $3,000$ travel offers.

Figure 4.5 shows the distribution of users grouped by age, education and their occupation. It can be observed that a user with midschool education cannot be in professional work or a worker cannot be a postgraduate. These were simple constraints that were added as a part of the data population process.

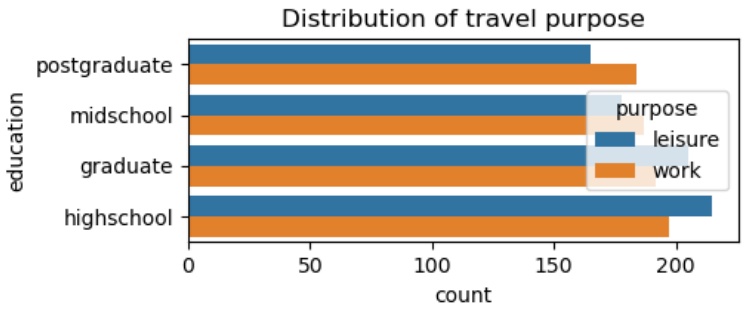

Figure 4.6: Distribution of users by travel purpose and education

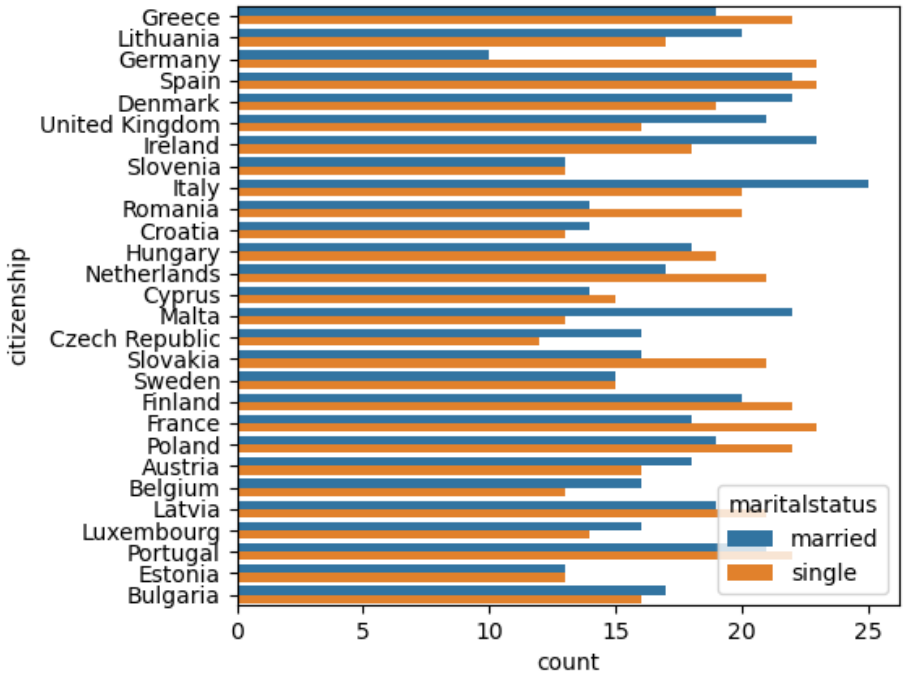

Figure 4.7: Distribution of users by country and marital status

Figure 4.7 shows the distribution of users by country and marital status. It covers countries all around the European Union. This would be useful in case of data analysis on country basis or suggestions specific to each country.

Figure 4.6 shows the distribution of users by travel purpose and education while Figure 4.8 shows the distribution of users by service and meal preference. These provide insight into the travel requests made by the travellers and the kind of preferences they had.

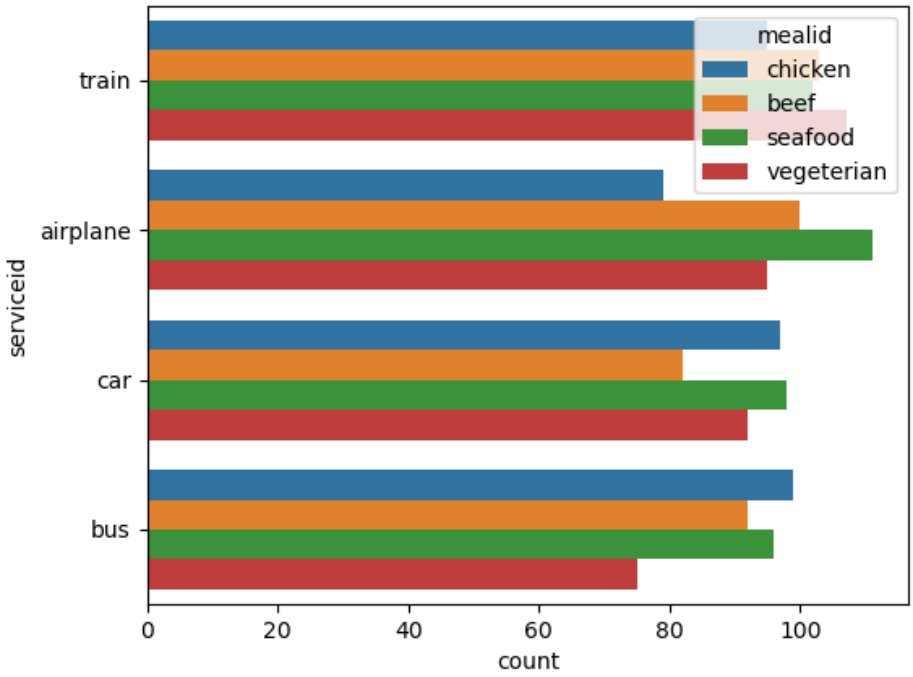

Figure 4.8: Distribution of users by service and meal preference

4.4 Knowledge Base

Recommender Systems are built with the foundation of a strong knowledge base of user's preferences and behavioral characteristics. It could be a model built from a training dataset or rules mined from historical data. The rules could be mined from the user's history or from other users with similar profile/preferences. In our work, we have chosen the latter. The dependency

on existence of historical data is known as the cold start problem. Our system overcomes this by mining association rules (Section 2.6) from a traveller dataset generated according to our requirements and specifications. The recommendation has to be handled at multiple situations in the system and the knowledge base should be accordingly diverse. Moreover, the representation and the data format of the rules play a vital role in the efficiency of the system. The ease and modularity corresponds to how quickly and accurately the results can be presented.

The association rules to be mined, need to address the following scenarios.

1. Travellers whose preferences is not known/specified
2. Travellers who do not have a past purchasing history to understand their travel choices

Also, this needs to be done based on the existing data in the system. The information available from the dataset is the static profile data, traveller preferences, health issues, travel choices and the facilities of corresponding travel offers.

The required format of such rules are:

1. **RULE TYPE 1**

   Profile $\implies$ Preferences

   Given a traveller's profile like age, gender, occupation we predict their preferences for service, meal etc.

2. **RULE TYPE 2**

   Preferences $\implies$ Offers

   Given a traveller's preferences in the request, we predict the type of offers (based on its facilities) that they will be interested in.

Both the rules together cover the major part of personalising the travel experience for a traveller. It suggests the possible preferences and possible travel choices that would be essential for a new user.

Once the type of rules were decided, the next step is to implement the association rule mining on the generated dataset. Different approaches were tried and tested before finalising on the process. Changes in approaches were

based on programming language, data preprocessing etc. The basic structure of the process remains the same.

- Table joins along with preprocessing to serve as input for the algorithm
- Generate rules using the algorithm
- Visualise them and validate the rules

Implementing the same algorithm in different programming languages could offer insight into the time taken by each language, the storage, simplicity of the syntax and also the results. The final result mainly differ in the template of the rules, the datatype in use ,quality metrics and their visualisation.

For our experiments, the two languages in discussion are Python and R programming. Both are well known for their simplicity of the language and specificity to data analysis and mining. Both the languages are open-source and have abundance of support from the programming community. While Python has been extremely popular for use in data processing, R has specialized packages for statistical computing.

The input in case of both the experiments shall remain the same, so that the results can be compared easily. The algorithm in use shall also be the same in both the experiments. Moreover, Python and R have respective modules for implementing association rule mining via apriori algorithm. The introduction to the language, function modules or libraries in use, the corresponding parameters, will be explained in the upcoming sections.

Experiment 1 - Python

Based on our research, Apriori algorithm (Section 2.7) was chosen to be used for association rule mining and the implementation to be done in Python.

The following Python modules were used

pandas is an open source tool for data manipulation, analysis and cleaning. It is well suited for different kinds of data, such as:

- tabular data with heterogeneously-typed columns
- ordered and unordered time series data
- arbitrary matrix data with row and column labels
- unlabelled data
- any other form of observational or statistical data sets

mlxtend.frequent_patterns[4] is a Python library of useful tools for data science tasks. In our work, it is used to retrieve frequent itemsets and association rules using apriori algorithm.

apriori(df, min_support*= 0.5, *use_colnames = False, max _ len = None, verbose,* = 0 *low_memory = False)

In the following we briefly explain the parameters.

- **df**: one-hot encoded pandas DataFrame as input (one-hot implies values can be true/false only)
- **min_support**: float (default: 0.5)
- **use_colnames**: bool (default: False) If True, uses the DataFrames' column names in the returned DataFrame instead of column indices.
- m**ax_len**: int (default: None) Maximum length of the itemsets generated.
- **verbose**: int (default: 0) Shows the number of iterations if ≥ 1 and low_memory is True. If = 1 and low_memory is False, shows the number of combinations.

[4]http://rasbt.github.io/mlxtend/

- **low_memory**: bool (default: False) If True, uses an iterator to search for combinations above min_support.

Returns pandas DataFrame with columns ['support', 'itemsets'] of all itemsets that are $\geq$ min_support and $<$ than max_len. Each itemset in the 'itemsets' column is of type frozenset, which is a Python built-in type that behaves similarly to sets except that it is immutable.

association_rules(df, metric = 'confidence', min_threshold $= 0.8$ ***, support_only = False)***

The parameters are explained as follows

- **df**: pandas DataFrame of frequent itemsets
- **metric**: string (default: 'confidence')

  Metric to evaluate if a rule is of interest.
- **min_threshold**: float (default: 0.8). Minimal threshold for the evaluation metric, via the metric parameter, to decide whether a candidate rule is of interest.
- support_only : bool (default: False). Only computes the rule support and fills the other metric columns with NaNs.

Returns pandas DataFrame with columns "antecedents" and "consequents" that store itemsets. The scoring metric columns are "antecedent support", "consequent support", "support", "confidence", "lift", "leverage", "conviction" of all rules for which metric(rule) $\geq$ min_threshold. Each entry in the "antecedents" and "consequents" columns are of type frozenset, which is a Python built-in type that behaves similarly to sets except that it is immutable

UserProfile:

In order to get the user's profile and preferences together in one table, *Traveller, Travel Request, Request, User Health, User Partner, Service preferences, Luggage Preferences and Meal preferences* were joined to create one table called *User Profile*. The timestamp was used to retrieve the preference closest to the request (for Rule type 1). In order to facilitate frequent itemset mining, preference options and age were converted to categorical data. For example, age < 30 as youngadult or age > 60 as elderly.

UserOffer:

In order to get the user's preferences and travel choices together, *Traveller, Travel Request, Request, User Health, User Partner, Service preferences, Luggage Preferences, Meal preferences, Travel Offer, Offer Detail* were joined to create one table called *UserOffer* . The facilities available in the different segments are concatenated together (for Rule type 2).

The created tables resembling the transactions are then fed into apriori algorithm. The following code snippet shows how the apriori algorithm is implemented in python using mlxtend library. The input to the algorithm is taken as csv, the data is first processed to fill up the empty fields and then encoded to suit the input format (i.e.) the allowed values are either 0/1 or True/False (one-hot encoding). The minimum support threshold for frequent itemset is taken as 10 percent. Frequent items are fetched using the *apriori()* and then the rules are mined using *association_rules()*.

RESULT

UserProfile data generated 4000+ rules and UserOffer generated 600+ rules but on evaluating them, it was observed that the rules do not follow the required format. In other words, the antecedents should only include features of the profile and consequents should reflect only the preferences for rule type 1 (Profile $\implies$ Preferences) whereas the antecedents and the consequents of the rule were arbitrarily picked up based on the frequency. Similarly in rule type 2.

Programmatically filtering out the rules by python parsing to follow the required format would not be a scalable solution with an evolving database and also not result in efficient set of rules for the knowledge base.

Code snippet of association rule mining in Python

```
import pandas as pd from mlxtend.frequent_patterns
import apriori, fpgrowth, association_rules
from mlxtend.preprocessing import TransactionEncoder


# Loading the Data
data = pd.read_csv('useroffer.csv')

# Transactions
basket = (data.fillna(0))

#data encoding apriori
data_encoded=pd.get_dummies(basket)

# Building the model
frq_items_ap = apriori(data_encoded, min_support=0.1,
use_colnames=True)


# Collecting the inferred rules in a dataframe
rules_ap = association_rules(frq_items_ap,
metric="lift",
min_threshold=1)

rules_ap = rules_ap.sort_values(
    ['confidence', 'lift'],
    ascending=[False, False])
```

Experiment 2 - R programming

R [5] is a language and environment for statistical computing and graphics. It was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R provides a wide variety of statistical and graphical techniques, and is highly extensible. It is also available as Free Software under the terms of the Free Software Foundation's GNU General Public License. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

Being simpler and faster to use, R was our immediate second approach to mine association rules from traveller data. The algorithm used remains unchanged - apriori, however, the results can be filtered and visualised in a more efficient manner.

arules [25] is a package that can be used to represent, manipulate and analyze transaction data and patterns (frequent itemsets and association rules). It differs from the association rule mining in Python with its ability to apply rule templates.

apriori(data, parameter=NULL, appearance=NULL, control=NULL)[6]

The parameters are explained as follows

- data - object of class '>transactions'
- parameter - object of class '>APparameter or named list.

  The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.8, maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).
- appearance - object of class '>APappearance or named list.

  With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.
- control - object of class '>APcontrol or named list. Controls the algorithmic performance of the mining algorithm

Returns an object of class '>rules or '>itemsets.

[5] https://www.r-project.org/

[6] https://www.rdocumentation.org/packages/arules/versions/1.6-6

In order to proceed with the association rule mining in R, the input required is a transactional database of traveller data. *UserProfile* and *UserOffer* tables created as a part of the Python Experiment were used as input. *UserProfile* to mine preferences with profile and *UserOffer* to mine offer choices with preferences. The tables can be read as a csv file and then converted into *'transactions'* class.

We have followed a minimum support of 10% and confidence of 50%. arules also allows to set the rule templates according to our required format using the 'appearance' parameter. Both the antecedents and consequents can be restricted to follow a pattern and those rules will be filtered out. The pattern can be a direct match filter or it could be specified by a regexp.

By default, arules allows only one item in the consequent. This could be changed by *minlen* parameter but for our system having one item as consequent makes it easier in the scoring process.

It also has a subset feature that can be used to filter out redundant rules as bigger rules can imply the smaller rules with fewer items. However, in our case, we have decided to keep all the rules as every match of rule consequent will be useful while ranking the offers. The mined rules can now be retrieved by *inspect()* to see the top few rules, or it could be visualised by plotting them using the *plot()*. The antecdents and consequents of the rules along with their quality metrics can also be retrieved separately as list or itemmatrix.

Code snippet of association rule mining in R

```
require(arules)
require(arulesViz)

#reading data
userofferdata <- read.csv(file='useroffer_new.csv')
offerdata <- as (userofferdata, "transactions")

userdata <- read.csv(file='userprofile.csv')
userdata <- as (userdata, "transactions")

#generating association rules
offerItems <- grep("^offer", itemLabels(offerdata),
value = TRUE)
prefItems <- grep("^pref", itemLabels(offerdata),
value = TRUE)
offer_rules <- apriori(offerdata,parameter
=list(support=0.1,confidence=0.5),
appearance =list(lhs=prefItems,rhs=offerItems))

uprefItems <- grep("id=[a-zA-Z]",
itemLabels(userdata), value = TRUE)
user_rules <- apriori(userdata,
parameter =list(support=0.1,confidence=0.5),
appearance =list(rhs=uprefItems))
```

RESULT

The rule template was applied successfully and the rules followed the target format. However on close observation, the rules did not suggest a lot of information. It provided information only on certain traveller preferences and features of the offer.

purpose=work $\Longrightarrow$ luggageid=bag

mstatus=single $\Longrightarrow$ accid=pet

age=elderly $\Longrightarrow$ accid=person

pref_serviceid=car $\Longrightarrow$ offer_accid=yes

This led to the realisation that only some of the features were always filled with data and most of them were missed out as the data population code had included NULL values to make it close to reality.

Modification

In order to generate a proper knowledge base filled with information, it is necessary that the data is complete and provides enough to mine rules that can provide suggestions on all features of the data items. Hence, the tables *UserOffe*r and *UserProfile* were further pre processed to include more traveller profiles and their travel choices and make the dataset suitable for association rule mining. The rerun resulted in enriched rules. The rules were better visualised by plotting them using R. The plot has various suggestions like 'matrix', 'mosaic', 'doubledecker', 'graph', 'paracoord', 'scatterplot', 'grouped matrix', 'two-key plot', 'matrix3D', 'iplots' etc. The choice of the plot is according to the application and in our scenario the 'graph' and 'paracoord' bring out the rule characteristics in the best way.

Observation

Observing the final rules help us validate the process of creating the knowledge base which in turn implies that in the presence of real historical traveller data, the predictions will be precise in guiding their recommendations and providing a personalised experience. These set of rules has to be updated periodically, to keep it updated with the incoming data and the period should be determined based on the system demand.

4.5 Ranking

Recalling our goal for the system, to be able to rank the travel offers such that it is ranked personally for the traveller. To achieve this, we use the rules mined in the previous step to rank the travel offers.

Ranking module can be split into two steps as follows.

1. Compare preferences mentioned in travel request and preferences present in the RULE antecedents to get the relevant rules.
2. Use the RULE consequents (facilities) to rank the list of available travel offers by how precise the facilities match

4.5.1 Preference Vector Comparison

From the knowledge base of mined rules, we need to pick out the relevant rules for the travel request. This gives the rule consequents required for the next step. Comparing the preferences mentioned in travel request and preferences present in the RULE antecedents is basically a comparison of two lists of preference items. The different cases are as follows.

- *Presence of complete vector* - The request preferences are completely present in the rule. It could be a single item or multiple items and their order does not matter. The presence of all the preference item implies that the rule is completely relevant and should be picked for the ranking process.
- *Partial Presence* - The request preference is partially present in the rule. It means the rule could include additional preference items or the request could have additional preference items. Nevertheless, the rule is partially relevant and can be picked for the ranking process based on how well they match. An indicator of relevance could be the match level between the preference vectors

Since the order does not matter and we need a comparison to be based on items, the best option is **Set Comparison** to retrieve the rules that are relevant to the request. The match level is also useful in picking only the top few relevant rules for scoring the offers later. In code, set comparison works by intersection and union to get match level.

Example of retrieving relevant rules for a travel request :

For a preference vector like:

("pref_healthid=visual", "pref_mealid=vegeterian", "pref_luggageid=bag"),

the following rule item is a complete match:

(pref_healthid=visual, pref_mealid=vegeterian) $\implies$ (offer_serviceid=airplane),

and the following rule item is a partial match with match level 1:

(pref_accid=pet,pref_mealid=vegeterian) $\implies$ (offer_facility=walking).

Conditions for set comparison: In order to implement the various scenarios of comparison discussed above, we need conditions based on which the relevant rules can be retrieved. These compare the rule items from the knowledge base and preference vector from the request and calculate how well they match.

Perfect match:

Matchlevel = length(PreferenceVector) = length(RuleItem)

Complete presence:

Matchlevel = length(PreferenceVector)

Partial:

Matchlevel=length(RuleItem)

Matchlevel > length(PreferenceVector)/2 AND (length(RuleItem) - Matchlevel)< length(PreferenceVector)/2

The following code snippet shows how the preference vector comparison with the knowledge base, to retrieve the top relevant rules was implemented in R. It follows a simple set comparison as mentioned earlier in Section 4.5.1.

```
#saving top few relevant rules with quality index and
match level in a dataframe
rulesdf=data.frame(Preference_Vector=character(),
Rule_Item=character(),
Match_level=numeric(),
Rule_Consequent=character(),
Confidence=numeric(),
Lift=numeric(),
stringsAsFactors = FALSE)

index<-0
df_index<-0
for (rule_item in offer_rules_ante)
{
  index<-index+1
  if(length(intersect(rule_item,request_pref)))
  {
    matchlevel=length(intersect(rule_item,request_pref))
    if ((matchlevel==length(request_pref)) ||
    (matchlevel==length(rule_item)) ||
    ((matchlevel>length(request_pref)/2)
    &&((length(rule_item)-matchlevel)<length(request_pref)/2)))
    {
      rulesdf[df_index,] <- list(list(request_pref),
      list(rule_item),
      matchlevel,offer_rules_conse[index],
      offer_rules_quality[index,2],
      offer_rules_quality[index,3])
      df_index<-df_index+1
    }

  }
}
```

4.5.2 Scoring List of Offers

The travel offers that have been received for the current request, have to be ranked. The top relevant rules that were picked in the earlier step will provide a set of rule consequents. These are basically suggestions for some facilities that should be present in the travel offers.

Moreover, the travel offer should match with the request as well. The traveller while providing the preferences in the request, can add an additional priority number. This priority is useful in overriding the suggestion provided by the rules and thus, preferences are more personalised than guided by suggestions made from popularity or past history.

Hence the scoring is based on a comparison between *rule consequents*, *preference given by user* and *facilities in the offer*.

Scoring strategy:

For each offer,

- Each rule consequent adds a point to matching offer multiplied by confidence (only when lift is greater than or equal to 1).

  $item\_score = rel * conf; lift \geq 1$

  rel is the match level between the rule consequent and offer. conf is the confidence of the rule in use.

  This is repeated with all the rule consequents to get a final *rule_score* by adding all *item_score*.

- Preference vector adds a point to matching offer multiplied by the priority.

  preference_score = rel * user_priority ;

  rel is the match level between the user preference and offer. user_priority is number mentioned by user.

- Calculate overall score to rank

  final_score = rule_score + preference_score

```
#comparing rule consequents and request preference to
available offers and scoring them
i=0
for (offer in avail_offers)
{
  score=0
  for (row in 1:nrow(rulesdf))
  {
    relevance=length(intersect(offer,
    rulesdf[row, "Rule_Consequent"]))
    lift=rulesdf[row, "Lift"]
    conf=rulesdf[row, "Confidence"]
    if(lift >1)
    {
      formula=relevance*conf
      score=score+formula
    }
  }
  prefscore=length(intersect(substring(offer,
  regexpr("=", offer) + 1),
  substring(request_pref, regexpr("=", request_pref) + 1)))
  pref_priority=sample(1:3, 1)
  print(offer)
  print(score+prefscore*pref_priority)
}
```

The code snippet shows how the scoring of potential travel offers was implemented in R. Each of the offers were compared with the top relevant rules and scored with the quality metrics of the corresponding rules.

Chapter 5

Experimental Evaluation

In this Chapter, we present our results obtained from the system. Section 4.4 detailed the Association Rule Mining Process (Experiments with Python and R) and Section 5.1 reports their corresponding results. Section 4.5 explained the Ranking module (Preference Vector Comparison and Scoring) and Section 5.2 reports the corresponding results.

5.1 Knowledge Base

As explained in Section 4.4, two join tables *UserProfile* and *UserOffer* were created and used as our database to mine association rules. Table 5.1 and Table 5.2 shows a snapshot of the same.

Table 5.1 includes the static profile information like *age, gender, occupation, mstatus and citizenship*, along with the traveller's preferences about *service, meal, luggage and their health issues*. This allows us to mine rules to predict the preferences given a user profile. *UserProfile* was created by taking each traveller data with their request and retrieving the corresponding preferences at the time of request by comparing timestamp of the request with the timestamp of the preferences and health issues. UserProfile data generated 4000+ rules and Table 5.3 shows a snapshot of the mined rules.

age	middleage	youngadult	elderly	middleage	elderly
gender	M	F	M	M	F
education	postgraduate	midschool	midschool	graduate	graduate
occupation	professional	worker	worker	prpfessional	business
mstatus	married	single	married	married	single
citizenship	Greece	Lithuania	Germany	Spain	Germany
healthID		walking			walking
accID	pet			person	
serviceID	car	airplane		train	car
mealID	beef	chicken		seafood	
luggageID	bag	bag		bike	bag
purpose	leisure	work	work	leisure	leisure

Table 5.1: Snapshot of data from UserProfile

pref_healthid			walking	walking	
pref_accid	pet	pet			person
pref_serviceid	car	bus	car		train
pref_mealid	beef	chicken		seafood	
pref_luggageid	bag	bag	bag		bike
offer_facility	walking	visual	deafness	walking	visual
offer_accid	yes	no	no	yes	yes
offer_serviceid	airplane	buscar	cartrain	car	bus
offer_mealid	beefchicken	chickenseafood	vegeterian	chicken	seafood
offer_luggageid	bag	bag	bag	bike	bikebag

Table 5.2: Snapshot of data from UserOffer

Table 5.2 includes information regarding the preferences of a traveller with every request along with the facilities of the offer that was finally chosen as the travel choice. This information was retrieved by taking preference information from request using timestamp for comparison. The offer details were retrieved based on the offer that was chosen for each request. Thus we have all information acquired by the system for each request and it helps us in mining rules that can predict the potential travel offer based on facilities, given the traveller's preferences. *UserOffer* generated 600+ rules and Table 5.4 shows a snapshot of the mined rules.

antecedents	{'healthid_pregnancy'}	{'accid_0', 'mstatus_single', 'mealid_0'}	{'accid_pet', 'luggageid_bag'}	{'occupation_worker', 'age_elderly'}
consequents	{'gender_F'}	{'gender_F'}	{'mstatus_married'}	{'mstatus_single'}
A_support	0.0715	0.099	0.103	0.08
C_support	0.519	0.519	0.493	0.506
support	0.071	0.065	0.065	0.05
confidence	1	0.655	0.63	0.62
lift	1.925	1.262	1.277	1.236

Table 5.3: Snapshot of results from mining rules with User Profile data.

antecdents	{'offer_luggageid_bike', 'offer_accid_yes'}	{'offer_serviceid_train'}	{'offer_facility_walking'}	{'pref_luggageid_0', 'pref_accid_pet'}
consequents	{'offer_mealid_vegeterian'}	{'offer_luggageid_bag'}	{'offer_mealid_vegeterian'}	{'pref_healthid_visual'}
A_support	0.083	0.091	0.162	0.126
C_support	0.187	0.168	0.187	0.489
support	0.053	0.056	0.096	0.074
confidence	0.637	0.614	0.592	0.588
lift	3.408	3.64	3.167	1.201

Table 5.4: Snapshot of results from mining rules with User Offer data.

Experiment with Python

Both the set of mined rules from Table 5.3 and Table 5.4 clearly depict that the rules are not exactly following the format that was discussed earlier. They provide lesser useful information than expected by the system. Although Python includes other options to mine association rules, with repeated trials, and modified parameters, the rule format still remains the same.

Experiment with R

As discussed in section 4.4, R programming provides better results, the rule templates were applied compatible to the system and the visualisations help in understanding the results with more clarity. The additional modification to the database (data pre-processing) resulted in enriched rules for the knowledge base.

Results Investigation

Both the experiments in Python and R used the same database as input and followed the same apriori algorithm. The usefulness of both the rules still rely on the data pre-processing, in order to fill up the database with diverse set of traveller information. The difference in results was mainly attributed to applying rule templates and quality metrics.

The apriori algorithm in Python was done by [43]. It is a straight forward implementation of the algorithm including machine learning and data science utilities and extensions. It differs from R in providing additional quality metrics that will be helpful in analysing the rules and prioritising them. The current implementation includes *antecedent support, consequent support, support, confidence, lift, leverage and conviction* as part of the results.

Investigating into how the apriori algorithm was built in R, it came to light that *arules* package in R [26], apart from providing an infrastructure for manipulating input data sets and for analyzing the resulting rules, also includes interfaces to two fast mining algorithms, the popular C implementations of Apriori and Eclat by Christian Borgelt [10]. This includes the *appearance* feature that allows the programmer to specify what is to be included or not in the rule antecedents and consequents thus applying rule templates.

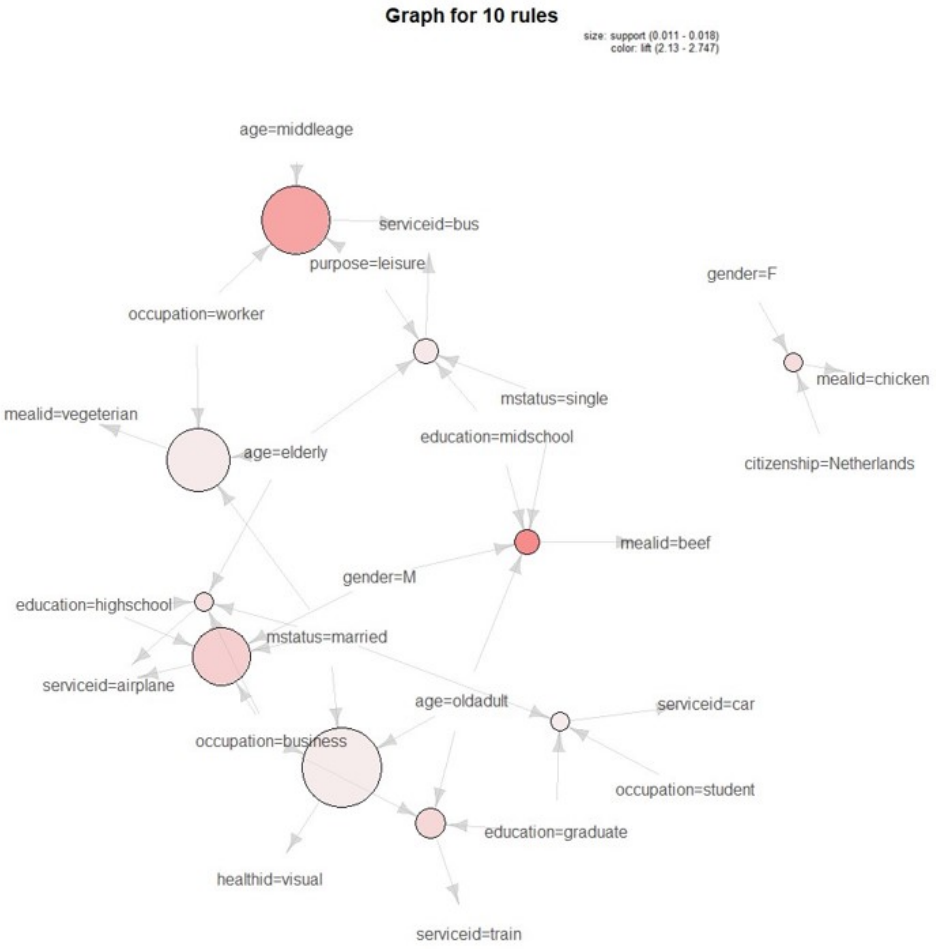

Figure 5.1: Association rules mined from User profile - Graph, Size of the nodes represent support and the color corresponds to lift.

In Figure 5.1 the top rules can be seen that, (age=middleage, occupation=worker, purpose=leisure implies serviceid=bus), (age=elderly, occupation=worker, mstatus=married implies mealid=vegeterian) and (age=oldadult, occupation=business, education=graduate implies serviceid=bus). Size of the nodes represent support and the color corresponds to lift.

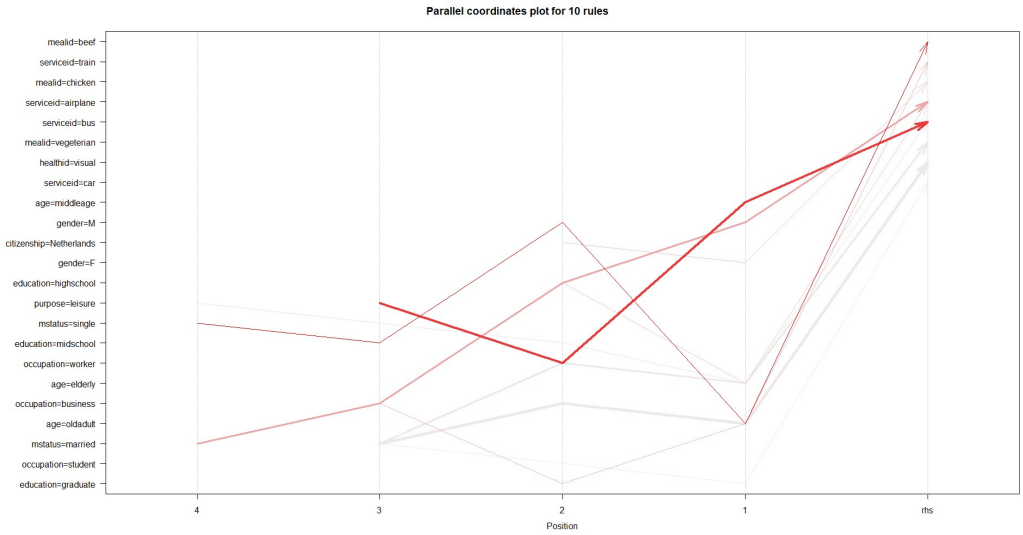

Figure 5.2: Association rules mined from User profile - Paracord.The width of the arrows represents support and the intensity of the color represent confidence.

In Figure 5.2 the top rules seen are (purpose=leisure, occupation=worker and age=middleage implies serviceid=bus) and (mstatus=married, occupation=business and purpose=leisure implies serviceid=airplane).The width of the arrows represents support and the intensity of the color represent confidence.

Figure 5.1 and Figure 5.2 are essentially talking about practically sensible user profiles and preferences. A middleaged worker travelling for leisure preferring a bus, An elderly married worker preferring vegeterian meal and a businessman preferring an airplane. The graph plot allows to visualise the rules with good readability while the paracord plot gives more insight into the support and confidence quality metrics.

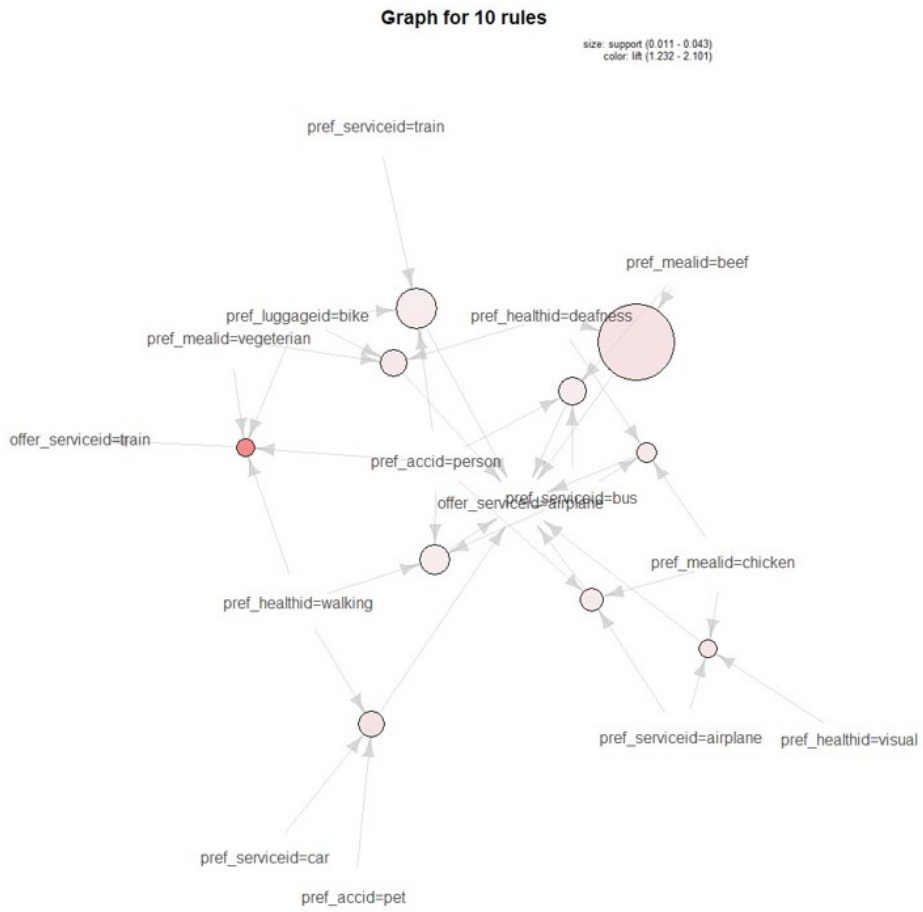

Figure 5.3: Association rules mined from User Offer - Graph.Size of the nodes represent support and the color corresponds to lift

In Figure 5.3 the top rules seen are (pref_serviceid=train, pref_luggageid=bike, pref_accid=person implies offer_serviceid=airplane) and (pref_healthid=visual, pref_mealid=chicken, pref_serviceid=airplane implies offer_serviceid = airplane). Size of the nodes represent support, the color corresponds to lift and 10 rules are picked for visual clarity.

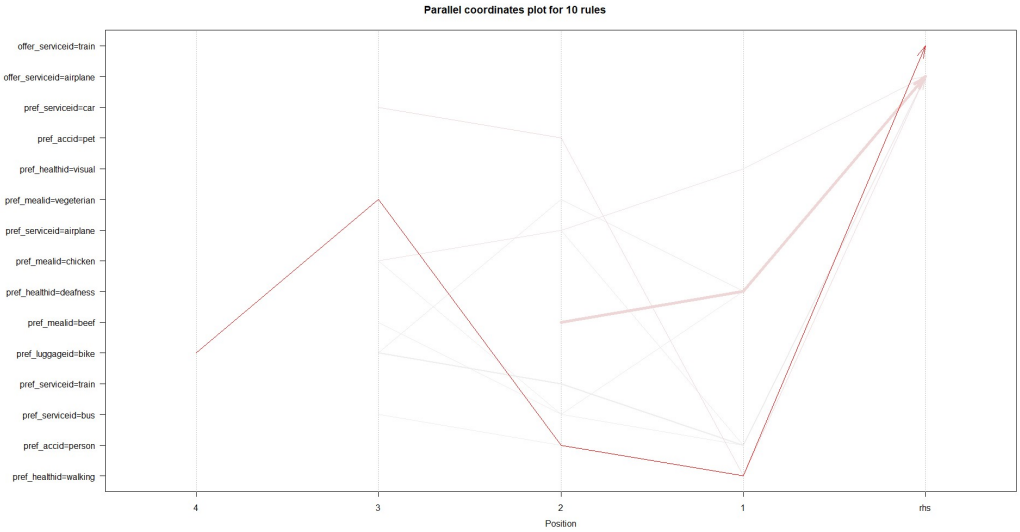

Figure 5.4: Association rules mined from User Offer - Paracord.The width of the arrows represents support and the intensity of the color represent confidence.

In Figure 5.4, the longest rule is (pref_luggageid=bike, pref_mealid=vegeterian, pref_accid=person, pref_healthid=walking implies offer_serviceid=train) and shortest is (pref_mealid=beef and pref_healthid=deafness implies offfer_serviceid =airplane)

These results in turn validate the learning process, and with additional real traveller information, the knowledge base can help in predicting and recommending the best travel offers suited to the traveller.

5.2 Ranking

The ranking module involved two steps: Preference Vector Comparison and Scoring.

In Section 4.5.1 we explained the conditions to pick out the top relevant rules from the knowledge base that can be later used for scoring. To demonstrate our example, we ran these conditions on the knowledge base. The result is shown in Table 5.5.These rule consequents give insights into the facilities that the potential offers should include in order to be best suited for the user.

In Section 4.5.2 we discussed the strategy for scoring the potential travel offers based on the preferences mentioned in the request along with the picked

rule	match level
pref_luggageid=bag $\Longrightarrow$ offer_luggageid=bag	1
pref_luggageid=bag $\Longrightarrow$ offer_facility=walking	1
("pref_healthid=visual", "pref_mealid=vegeterian") $\Longrightarrow$ offer_luggageid=bag	2
("pref_healthid=visual", "pref_mealid=vegeterian", "pref_luggageid=bag") $\Longrightarrow$ offer_serviceid=airplane	3
("pref_healthid=walking", "pref_mealid=vegeterian", "pref_luggageid=bag") $\Longrightarrow$ offer_serviceid=airplane	2

Table 5.5: Snapshot of results from picking relevant rules with match level.

out rule consqeuents. The overall calculated score is then used to rank the travel offers that can be presented to the user.

A sample run with a travel request and a set of potential travel offers were ranked by the system and Figure 5.5 shows the results. It includes intermediate results of the relevant rules picked from the knowledge base and the final list of ranked offers. The ranking in this example is mainly influenced by the rules mined and stored in the knowledge base.

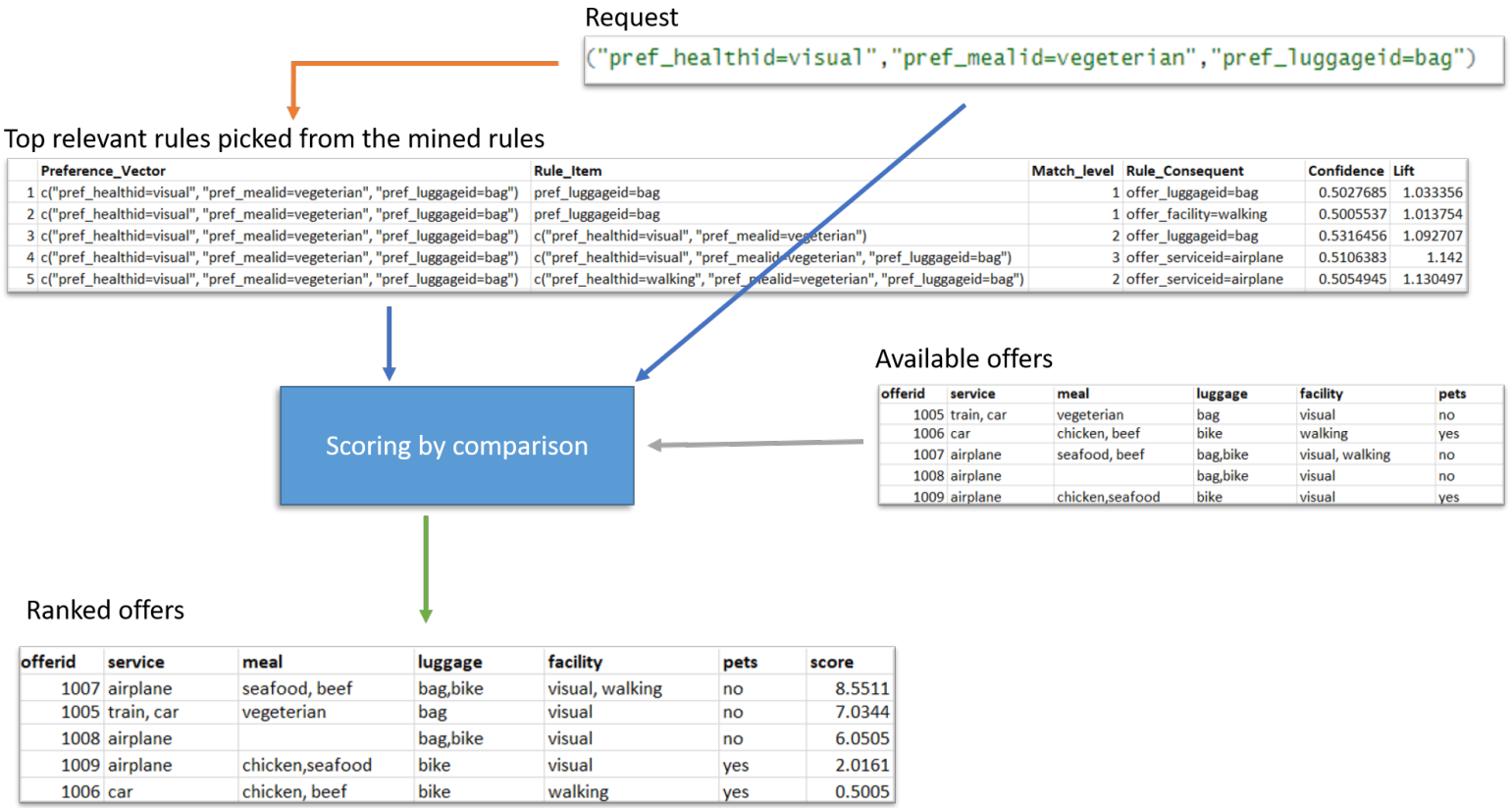

Figure 5.5: Sample run of the ranking module that returns the ranked offers

5.3 Prototype

In this section, we present our solution in a demonstrative manner, from the eyes of a traveller. We have built a prototype to envision the system, in order to understand the input taken from a user and how the results are presented in a personalised manner to suit his/her requirement.

Figure 5.6 shows the home screen of the *Travel Companion*. The traveller starts the travel journey from this screen. The input required in this screen is the main parts of a travel request. The kind of information that is unique to each request (i.e.) Source, Destination, Travel Date and Purpose.

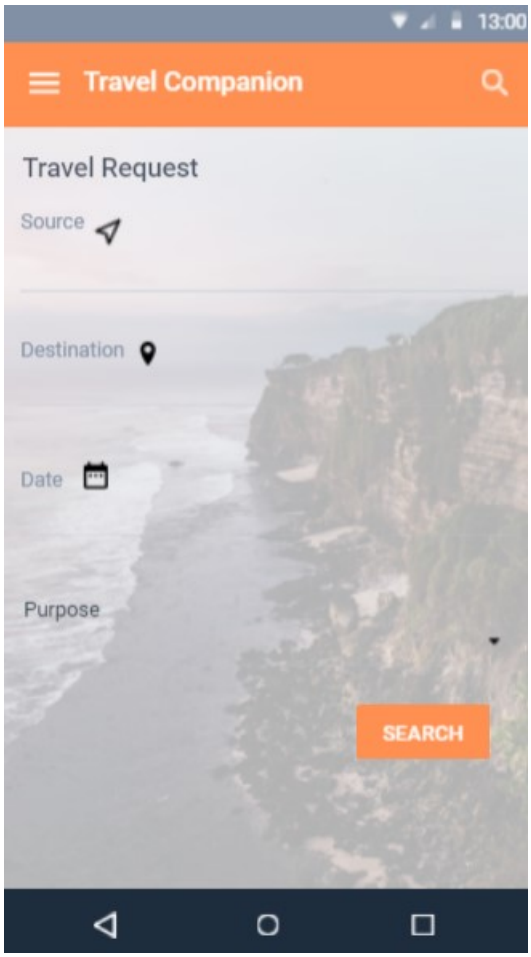

Figure 5.6: The prototype of the Travel Companion: Home Screen

Figure 5.7 shows a sample filled out travel request. A trip from Milan to Bratislava on 10th of October, 2020 for leisure.

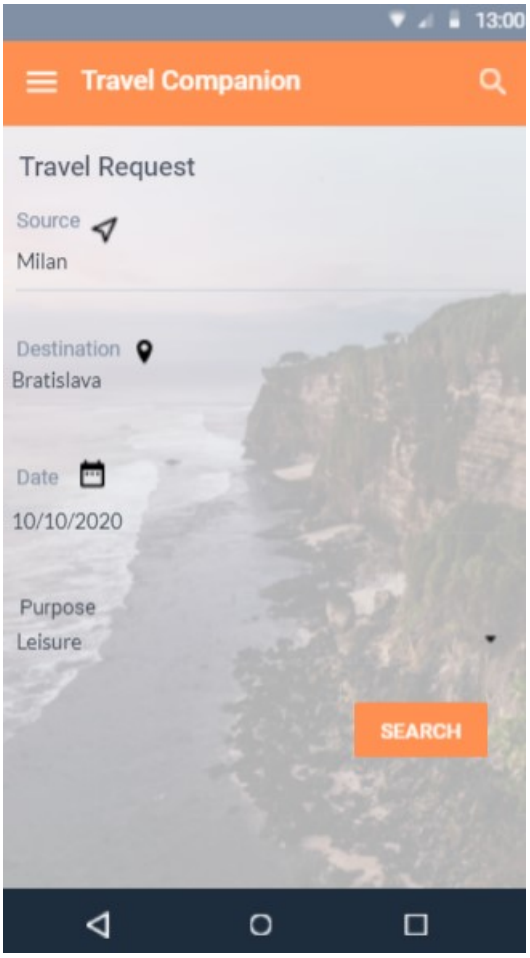

Figure 5.7: The prototype of the Travel Companion: Filled Travel Request

Figure 5.8 shows the side panel navigation bar. This shows the profile of the traveller and other links to navigate. The Preferences link is of most importance to our solution. To fetch the user preferences which helps us in the personalization process.

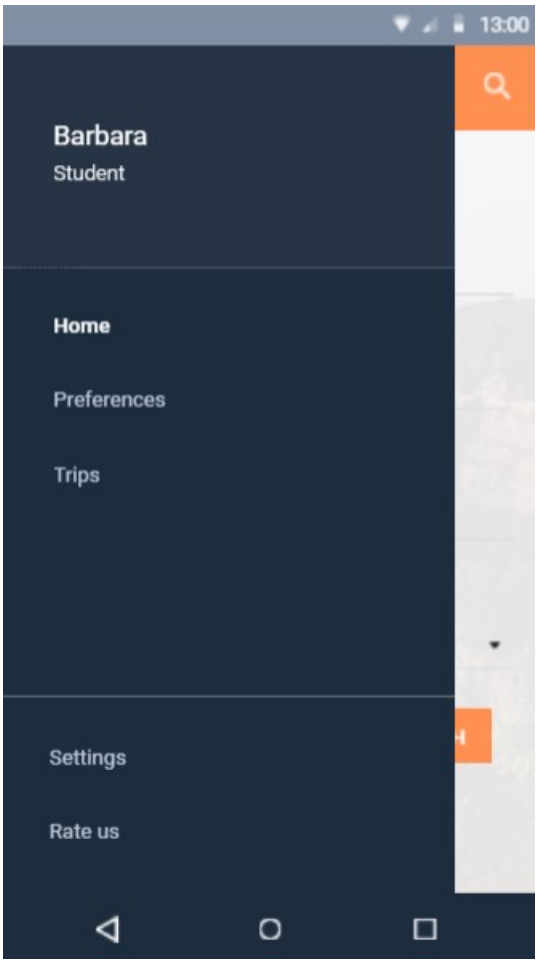

Figure 5.8: The prototype of the Travel Companion: Side Panel Navigation

Figure 5.9 shows the form to be filled up for user preferences. The user mentions health issues, service, meal and luggage preferences etc. These are saved until further change is made and henceforth will be taken into consideration for filtering the results presented to the user.

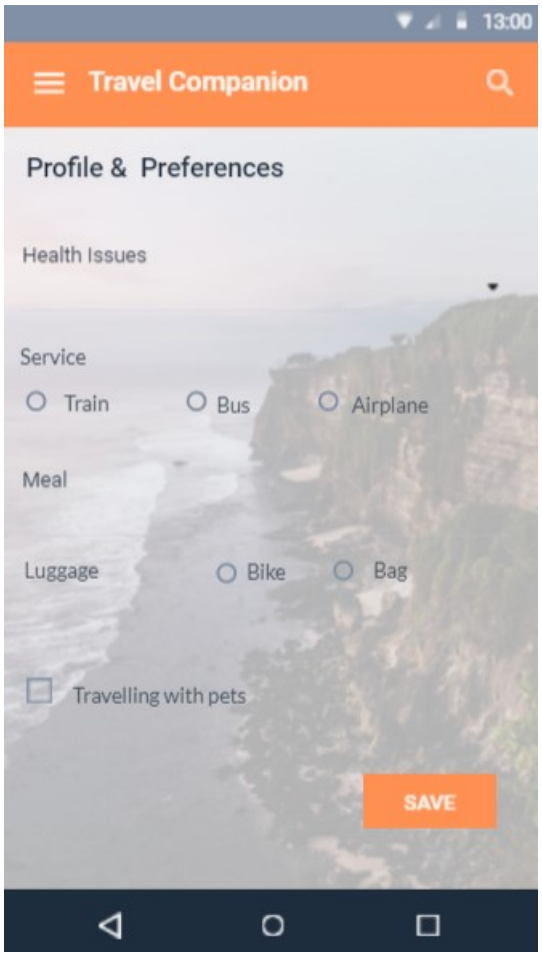

Figure 5.9: The prototype of the Travel Companion: Preferences screen

Figure 5.10 shows a sample filled out preferences screen. The user has visual disabilities, prefers an airplane, vegetarian meal and has luggage requirements of a bag.

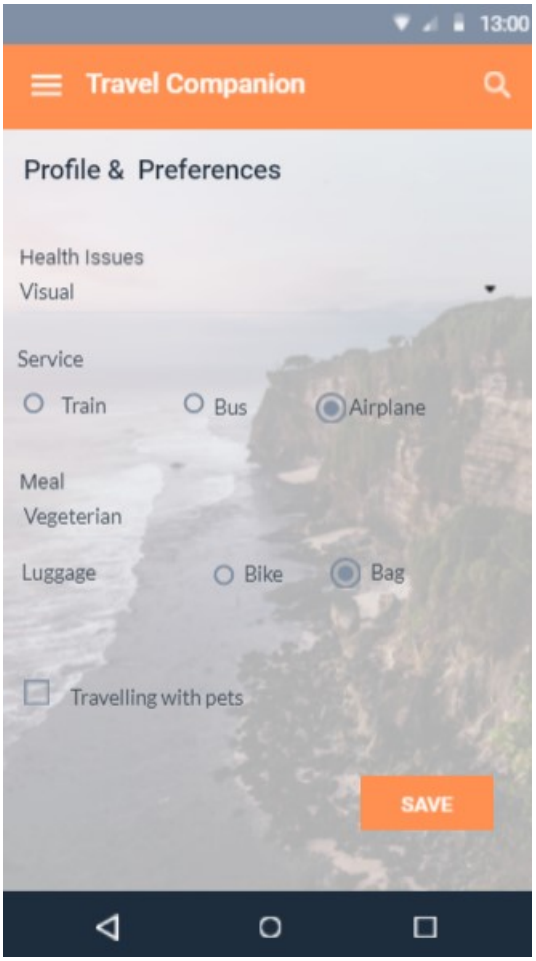

Figure 5.10: The prototype of the Travel Companion: Filled Preferences

Figure 5.11 shows the list of travel offers for the travel request made. They are ranked according to the preferences made by the user and recommendations from the system knowledge base. The icons on the left depict the mode of transport and the icons on the left shows the special facilities available in the offer. It also includes the allowed luggage and meal availability. The *paw* icon symbolises pets being allowed in the journey and the green leaf symbolises Eco friendliness of the offer.

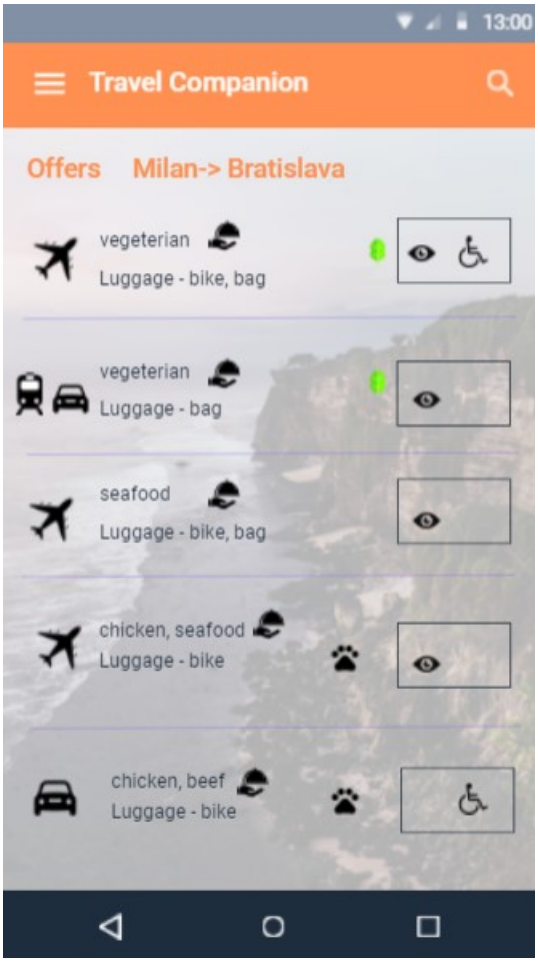

Figure 5.11: The prototype of the Travel Companion: Travel Offers

Figure 5.12 shows the detailed offer information that can be seen when a user clicks on a particular offer. The sample is shown for the second offer on the offers page. It includes information regarding stopovers and the icons transformed into words for clarity, helping the user make their travel choice.

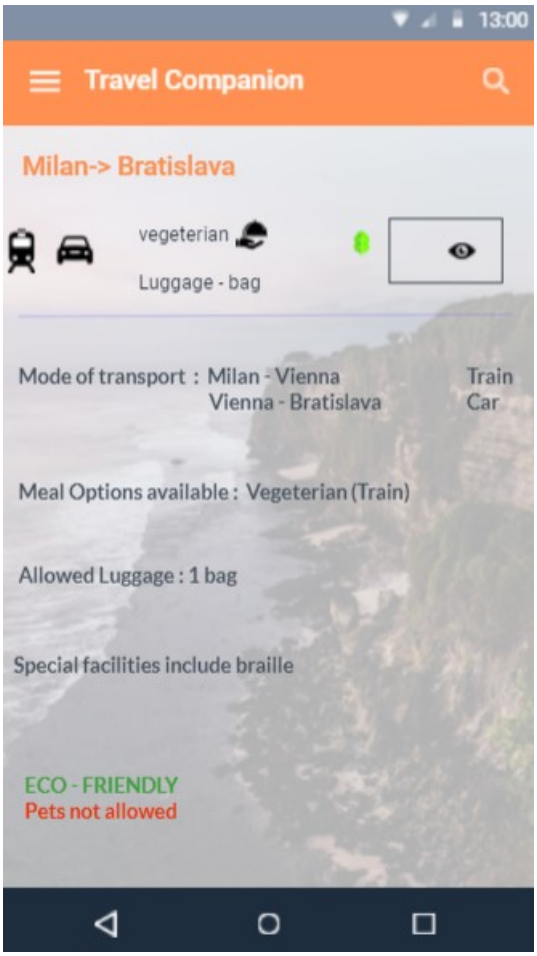

Figure 5.12: The prototype of the Travel Companion: Detailed Offer Information

Chapter 6

Conclusions

In this thesis, we presented our solution to build a data mining based recommender system that provides a ranked list of travel offers, filtered and personalised by the user's context and preferences.

In particular, Chapter 2 gave an introduction to the Travel Companion, the theoretical concepts that the system is based on and the techniques like frequent itemset, association rule mining, used for the same. It explained the idea of recommender system and its various types. Chapter 3 discussed the state of art methodologies working with context aware systems. It briefed on relevant interesting papers in the area of data mining and recommendation. Chapter 4 presents the system, provides an overview of the various steps involved and then explains each in detail starting from Database Design, Data population, Building the knowledge base and Ranking module. Chapter 5 then reports the results of the experiments discussed in Chapter 4 and evaluates them to validate the solution.

Discussion: From our experimental evaluation, it is observed that the solution for the system presented is indeed suitable and can be further explored to enhance its feature set. The demonstration of the rule based knowledge model with the custom dataset shows that, on replacing with real data, the algorithm would continue to learn the user's preferences and provide personalised suggestions. The framework could be followed with enriched modules as addition.

Future work: In this section, we introduce some of the possible future works that would probably increase the application scope.

Data enrichment: To expand the size of dataset by populating the travel offers table with many features for each transportation mode. As a result of this expansion, comes the need for applying appropriate feature selection techniques to reduce the complexity of the mining process [11, 42].

Feature Extension: Another area of potential work is working on the levels of abstraction and adding more options to the travel request essentially making the process more user friendly.

Testing: Field testing with frequent travellers for usability to identify breaking points and any parameters to increase user engagement.

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Using data mining methods to build customer profiles. *IEEE Computer*, 34:74–82, 03 2001.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. *Mining Association Rules Between Sets of Items in Large Databases, SIGMOD Conference*, volume 22, pages 207–. 06 1993.

[3] Joshua Akehurst, Irena Koprinska, Kalina Yacef, Luiz Pizzato, Judy Kay, and Tomasz Rej. Explicit and implicit user preferences in online dating. volume 7104, pages 15–27, 05 2011.

[4] Narimel Bendakir and A Esma. Using association rules for course recommendation. 2006.

[5] Cristiana Bolchini, C Curino, E Quintarelli, FA Schreiber, and L Tanca. Context-addict. *Technical Report 2006.044*, 2006.

[6] Cristiana Bolchini, Carlo Curino, Giorgio Orsi, Elisa Quintarelli, Rosalba Rossato, Fabio Schreiber, and Letizia Tanca. And what can context do for data? *Communications of the ACM*, 52:136–140, 11 2009.

[7] Cristiana Bolchini, Carlo A Curino, Elisa Quintarelli, Fabio A Schreiber, and Letizia Tanca. A data-oriented survey of context models. *ACM Sigmod Record*, 36(4):19–26, 2007.

[8] Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. Context information for knowledge reshaping. *International Journal of Web Engineering and Technology*, 5(1):88–103, 2009.

[9] Cristiana Bolchini, Elisa Quintarelli, and Letizia Tanca. Carve: Context-aware automatic view definition over relational databases. *Information Systems*, 38:45–67, 03 2013.

[10] Christian Borgelt. Finding association rules/hyperedges with the apriori algorithm. 1996.

[11] A. Brankovic, M. Hosseini, and L. Piroddi. A distributed feature selection algorithm based on distance correlation with an application to microarrays. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(6):1802–1815, 2019.

[12] Sven Buchholz, Thomas Hamann, and Gerald Hubsch. Comprehensive structured context profiles (cscp): Design and experiences. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, pages 43–47. IEEE, 2004.

[13] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[14] Silvia Canale, Alessandro Di Giorgio, F Lisi, Martina Panfili, L Ricciardi Celsi, Vincenzo Suraci, and F Delli Priscoli. A future internet oriented user centric extended intelligent transportation system. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 1133–1139. IEEE, 2016.

[15] Alessio Carenini, Ugo Dell'Arciprete, Stefanos Gogos, Mohammad Mehdi Pourhashem Kallehbasti, Matteo Rossi, and Riccardo Santoro. ST4RT – semantic transformations for rail transportation. In *TRA 2018*, pages 1–10, 2018.

[16] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.*, pages 258–267. IEEE, 2004.

[17] Joëlle Coutaz, James L Crowley, Simon Dobson, and David Garlan. Context is key. *Communications of the ACM*, 48(3):49–53, 2005.

[18] Florian Daniel, Maristella Matera, Elisa Quintarelli, Letizia Tanca, and Vittorio Zaccaria. *Context-Aware Access to Heterogeneous Resources Through On-the-Fly Mashups*, pages 119–134. 01 2018.

[19] Toon De Pessemier, Jeroen Dhondt, Kris Vanhecke, and Luc Martens. Travelwithfriends: a hybrid group recommender system for travel destinations. In *Proceedings of the Workshop on Tourism Recommender Systems, in conjunction with the 9th ACM Conference on Recommender Systems*, pages 51–60, 2015.

[20] Roberto De Virgilio and Riccardo Torlone. A general methodology for context-aware data access. In *Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access*, pages 9–15, 2005.

[21] Roberto De Virgilio, Riccardo Torlone, and GJPM Houben. A rule-based approach to content delivery adaptation in web information systems. In *7th International Conference on Mobile Data Management (MDM'06)*, pages 21–21. IEEE, 2006.

[22] Patrick Fahy and Siobhan Clarke. Cass–a middleware for mobile context-aware applications. In *Workshop on context awareness, MobiSys*. Citeseer, 2004.

[23] A. Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. *Group Recommender Systems - An Introduction*. 05 2018.

[24] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and computer applications*, 28(1):1–18, 2005.

[25] Michael Hahsler, Christian Buchta, Bettina Gruen, and Kurt Hornik. *arules: Mining Association Rules and Frequent Itemsets*, 2020. R package version 1.6-6.

[26] Michael Hahsler, Bettina Gr, and Kurt Hornik. Introduction to arules — mining association rules and frequent item sets. 2006.

[27] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *International Conference on Pervasive Computing*, pages 167–180. Springer, 2002.

[28] Jon Herlocker, J.A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. *Proc. of CSCW 2000*, pages 241–250, 01 2000.

[29] D. Herzog and W. Wörndl. A travel recommender system for combining multiple travel regions to a composite trip. *CEUR Workshop Proceedings*, 1245:42–47, 01 2014.

[30] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Systems with applications*, 36(4):8509–8522, 2009.

[31] Marjan Hosseini, Safia Kalwar, Matteo Rossi, and Mersedeh Sadeghi. Automated mapping for semantic-based conversion of transportation data formats. In *Proceedings of the International Workshop On Semantics For Transport (Sem4TRA)*, volume 2447 of *CEUR-WS*, pages 1–6, 2019.

[32] Alireza Javadian Sabet, Matteo Rossi, Fabio A. Schreiber, and Letizia Tanca. Context awareness in the travel companion of the shift2rail initiative. In *Italian Symposium on Advanced Database Systems*, pages 199–206, 2020.

[33] Alireza Javadian Sabet, Matteo Rossi, Fabio A. Schreiber, and Letizia Tanca. Towards learning travelers' preferences in a context-aware fashion. In *International Symposium on Ambient Intelligence*. Springer, 2020.

[34] Manasawee Kaenampornpan, Eamonn O'Neill, and B Ba Ay. An integrated context model: Bringing activity to context. In *Proc. Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.

[35] Sumitkumar Kanoje, Sheetal Girase, and Debajyoti Mukhopadhyay. User profiling trends, techniques and applications. *arXiv preprint arXiv:1503.07474*, 2015.

[36] Antonio Miele, Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. A data-mining approach to preference-based data ranking founded on contextual information. *Information Systems*, 38:524–544, 06 2013.

[37] Aris M Ouksel. In-context peer-to-peer information filtering on the web. *ACM SIGMOD Record*, 32(3):65–70, 2003.

[38] B Pernici. Mais: Multichannel adaptive information systems. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 303–305. IEEE, 2003.

[39] Saverio Perugini, Marcos Gonçalves, and Edward Fox. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23, 06 2002.

[40] Danny Poo, Brian Chng, and Jie-Mein Goh. A hybrid approach for user profiling. In *36th Annual HICSS, 2003. Proceedings of the*, pages 9–pp. IEEE, 2003.

[41] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In *European Symposium on Ambient Intelligence*, pages 148–159. Springer, 2004.

[42] K Rajeswari. Feature selection by mining optimized association rules based on apriori algorithm. *International Journal of Computer Applications*, 119(20), 2015.

[43] Sebastian Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), April 2018.

[44] Taha Rashidi, Alireza Abbasi, Mojtaba Maghrebi, Samiul Hasan, and Travis Waller. Exploring the capacity of social media data for modelling travel behaviour: Opportunities and challenges. *Transportation Research Part C: Emerging Technologies*, 75:197–211, 02 2017.

[45] Ioanna Roussaki, Maria Strimpakou, Nikos Kalatzis, Miltos Anagnostou, and Carsten Pils. Hybrid context modeling: A location-based scheme using ontologies. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, pages 6–pp. IEEE, 2006.

[46] M Sadeghi, Petr Buchníček, A Carenini, O Corcho, S Gogos, M Rossi, R Santoro, et al. Sprint: Semantics for performant and scalable interoperability of multimodal transport. In *8th Transport Research Arena TRA 2020*, pages 1–10, 2020.

[47] J Sastrt and V. Suresh. A survey paper on frequent itemset mining. *Journal of Physics: Conference Series*, 1228:012050, 05 2019.

[48] Harini Sridharan, Hari Sundaram, and Thanassis Rikakis. Computational models for experiences in the arts, and multimedia. In *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, pages 31–44, 2003.

[49] Daniela Petrelli Elena Not Carlo Strapparava and Oliviero Stock Massimo Zancanaro. Modeling context is like taking pictures.

[50] Maria A Strimpakou, Ioanna G Roussaki, and Miltiades E Anagnostou. A context ontology for pervasive service provision. In *20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06)*, volume 2, pages 5–pp. IEEE, 2006.

[51] K. Venkatesh and J. Jabez. Recommendation and ranking of travel package to tourist. *Contemporary Engineering Sciences*, 8:401–407, 01 2015.

[52] Xindong Wu, Vipin Kumar, Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, G. Mclachlan, Shu Kay Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14, 12 2007.

[53] Stephen JH Yang, Angus FM Huang, Rick Chen, Shian-Shyong Tseng, and Yen-Shih Shen. Context model and context acquisition for ubiquitous content access in ulearning environments. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, volume 2, pages 78–83. IEEE, 2006.