UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA
(FINAL PROJECT PAPER)

MOBILNA APLIKACIJA ZA VODENJE POPOTNIKOV Z UPORABO PRIPOROČILNEGA SISTEMA (MOBILE APPLICATION FOR TRAVEL GUIDANCE USING A RECOMMENDER SYSTEM)

PREDRAG ZVEZDAKOSKI

2021
ZAKLJUČNA NALOGA (FINAL PROJECT PAPER)
ZVEZDAKOSKI

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga
(Final project paper)

Mobilna aplikacija za vodenje popotnikov z uporabo priporočilnega sistema

(Mobile application for travel guidance using a recommender system)

Ime in priimek: Predrag Zvezdakoski
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Branko Kavšek
Somentor: doc. dr. Vida Groznik

Koper, september 2021

Ključna dokumentacijska informacija

Ime in PRIIMEK: Predrag ZVEZDAKOSKI

Naslov zaključne naloge:
Mobilna aplikacija za vodenje popotnikov z uporabo priporočilnega sistema

Kraj: Koper

Leto: 2021

Število listov: 36 Število slik: 6 Število tabel: 4

Število referenc: 8

Mentor: doc. dr. Branko Kavšek

Somentor: doc. dr. Vida Groznik

Ključne besede: priporočilni sistem, mobilna aplikacija, vodnik

Izvleček:
Glavna tema diplomskega dela je predstavitev izgradnje aplikacije ki vsebuje priporočilni sistem za hotele, hostle, restavracije in aktivnosti v Sloveniji. Najprej predstavimo kaj so priporočilni sistemi skupaj z nekaterimi tipi tovrstnih sistemov. Nato predstavimo, kako smo podatke pridobili s spletnega mesta TripAdvisor, jih očistili in prilagodili, da so primerni za uporabo v naši aplikaciji. V osrednjem delu predstavimo, kako smo implementirali hibridni priporočilni sistem, ki primerja ocene posameznih primerov in ključne besede z minimalnimi uporabnikovimi preferencami.

Key document information

Name and SURNAME: Predrag ZVEZDAKOSKI

Title of the final project paper:
Mobile application for travel guidance using a recommender system

Place: Koper

Year: 2021

Number of pages: 36 Number of figures: 6 Number of tables: 4

Number of references: 8

Mentor: Assist. Prof. Branko Kavšek, PhD

Co-Mentor: Assist. Prof. Vida Groznik, PhD

Keywords: recommender system, mobile application, tour guide

Abstract:
The main topic of the thesis is describing the implementation of an application using a recommender system for Hotels, Hostels, Restaurants and “Things to Do” in Slovenia. First, we introduce the notion of a recommender system along with a few types of systems. Afterwards we discuss how we collected the data from TripAdvisor, cleaned it, and modified it to be usable in our application. The main part describes how we implemented a hybrid recommender system that compares all individual item ratings and keywords to the preferences of the user.

ACKNOWLEDGEMENTS

I would like to thank the University of Primorska and Famnit for accepting me as their student and providing help and support throughout my studies.
I would also like to express my deepest gratitude to my mentor Branko Kavšek and my co-mentor Vida Groznik for their involvement in the thesis. Their willingness to help during this pandemic enabled me to finish my studies. Secondly, without their support and guidance during each step of the way this thesis would not be possible.
Finally, I would like to thank my family for all the support and encouragement that they have given me.

Predrag Zvezdakoski
August 9, 2021

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The thesis describes the implementation of a recommender system. A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item [7]. The system uses the interactions between the user and the content to make item predictions. It saves data based on past user behavior, and then using this data, recommends items that would be of the user's interest.

Today, recommender systems are widely used throughout the internet. These systems increase the user experience. Showing relevant content to each user. This makes it easier and quicker for the user to discover content he would otherwise not see, or it would take him more time to manually look for his specific interests. Some popular examples utilizing recommendations include:

- Videos, music (YouTube, Spotify)
- Movies / series (Netflix)
- Online stores (Amazon, eBay)
- Social media (Facebook)

In the thesis, we present an application designed to recommend hotels, hostels, restaurants and "things to do" to the user. The recommendations presented are located in Slovenia. It is designed to be useful for both residents and tourists visiting the country. The application offers the possibility to select preferences for each of the categories and then get recommendations based on the information provided. This is useful for people who are visiting the country and cannot rate what they like there, since they are not familiar with the local offer. On the other hand, users who already know which restaurants or hotels they like, have the ability to rate them to further improve their recommendations, in addition to setting their initial preferences.

There are several approaches to how this data can be used by the system to make the item predictions. In this paper, a knowledge-content based hybrid recommender system will be described, which is used to recommend items to the user.

Hybrid recommender systems combine two or more recommendation strategies in different ways to benefit from their complementary advantages [4].

Knowledge-based recommender systems are a specific type of system that are based on explicit knowledge about the item attributes, user preferences, and recommendation criteria (which item should be recommended in which context) [1]. These systems are applied in scenarios where alternative approaches such as collaborative filtering and content-based filtering cannot be applied.

Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features [4]. In this system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes.

In Chapter 2, we provide insight into related work. We present some papers and their solutions to the problem we are working on. Chapter 3 presents the process of obtaining the data we need for the application. Chapter 4 describes how our recommender system works. In Chapter 5, we explain the implementation of the system as an android mobile application. We give our conclusions in Chapter 6.

2 RELATED WORK

A lot of work has been done in the field of tourist recommendations using various approaches. First, there are some papers, which show methods of recommending items based on explicit user input, and creating an algorithm, which makes predictions based on the data entered by the user and the item attributes. Comparing the input from the user with the item characteristics and finding the best matches to be used as the recommendations.

In his paper [3], Burke presents a hybrid collaborative filtering-knowledge based system. A collaborative filtering recommender system recommends to the active user the items that other users with similar tastes liked in the past [7]. The similarity in the taste of two users is calculated using the similarity in the rating history of the users. The example explained in the paper is intended to be used in e-commerce, nevertheless it shows that the combination of these two approaches reduces the weakness of the collaborative-filtering recommendations. The knowledge part of the recommender provides initial data collected from the user. Before showing predictions, the user is presented with some form of a questionnaire or preferences selection process. This way he can tell the system what he likes; this can then be used as a starting point to recommend items. Which is a much better starting point compared to not having any initial preferences selection and no data on the user. When the user uses the system more frequently and enough data has been collected, the system can start using collaborative filtering to improve the recommendations.

Zhang [8] proposes the use of preferences. Similar to the explanation in the previous paragraph, before enough data is collected from the active user to switch to a collaborative-filtering system, they use the preferences as a way to make item predictions. Another possibility shown is selecting the intent for an individual trip. That is, selecting criteria for the next trip that the user wants to make. Possibly, a trip with different preferences to those we have collected so far in the user model. Usually, these one-time searches do not have much influence on the user's preferences. This can be useful in cases where an upcoming trip is significantly different from previous trips the user has made (example: a user who often travels for work decides to go on holiday with his family). The main recommender system used by Park [6] is a choice between the primary recommender, collaborative-filtering and content-based, used in the cases where data is sparse and the recommendations from the first system would not be able to offer good predictions.

Next, there are approaches that use context, and review user comments to gain information not provided by the user, to increase the accuracy of the predictions. Some examples for information that can be gained through user comments is finding what the user values in his trips, this could include often reviewing the services or hospitality provided by a hotel, or the sightseeing opportunities for a trip.

Hariri [5] introduces a context aware recommender system that obtains contextual information by mining user reviews and combining them with user rating history to compute

a utility function over a set of items. Park [6], along with several other papers that use a similar idea, suggest using GPS on the user's mobile phone to collect information. This enables the system the capability to offer more relevant recommendations to each user, based on the user's physical location. That is, offering recommendations that are in a certain radius distance from the current location of the user.

3 DATA

The data consists of four different items that can be recommended to the user. They include hotels, hostels, restaurants and "things to do". Data for each category was scraped from TripAdvisor.

To find the needed items, we typed 'Slovenia' in the search bar. Then using the predefined tabs of the website labeled 'Hotels', 'Restaurants' and 'Things to Do' we can find a list of approximately 1000 items for each. Using the 'Selenium'[1] library in Python, we scraped all the information mentioned above for each item. The scraping algorithm can be found at https://github.com/Predrag10/tripadvisor-scrapers.

The data used from TripAdvisor included the following attributes for each type of item:

- *Hotel*: name, location, class, number of reviews, average rating, location rating, cleanliness rating, service rating, value rating, description, amenities
- *Restaurant*: name, location, number of reviews, average rating, price range (36%), cuisine, food rating, service rating, value rating, atmosphere rating
- *Thing to Do*: name, location, number of reviews, average rating, keywords based on user comments

The initial data collected had the following number of items in each category: Hotels (1005), Restaurants (997), and "Things to Do" (1020). We removed the items not matching our requirements. First, we removed all items, which were missing essential attributes (name or location). The next step was removing all items located outside of Slovenia. Lastly, we divided the hotels into two groups: Hotels and Hostels. All items which did not have a 'Hotel Class' attribute, which represents the number of stars a hotel has, were moved to the 'Hostel' group. The number of items remaining in each category are Hotels (385), Hostels (515), Restaurants (955), and "Things to Do" (391).

For the feature of searching items by region, using the location scraped, an attribute 'Region' was added. The region is defined based on the postal code of the location of the item. The user can choose among nine options when searching for recommendations: Slovenia or one of its eight regions. Currently, this groups several cities into one region. It could be changed later on to have more regions, offering more precise recommendations based on location. Or, the phone GPS could be used to recommend items that are the closest to the user.

To create a better content-based recommender system we used the 'Amenities' / 'Cuisine' attribute and divided them into keywords. This would enable the user to select his preferences, and allow us to use his ratings, as he uses the application, to tweak his inclination for each of the keywords, as well as all the ratings for each item.

The division of keywords was done by using the most frequent keywords in each category as the main keywords. Then, grouping the remaining keywords as sub-items to these main

[1] https://github.com/SeleniumHQ/selenium

keywords. There are exceptions to the 'most frequent keywords' rule. Using our judgement, in some cases we decide that a keyword with lower frequency better describes the item group, and then, that keyword is used as a main keyword instead.

3.1 Hotel keywords

Table 1 below shows the keywords we have selected and are used for describing hotel amenities, along with all the sub-keywords that are contained within them. The 'number of hotels' column provides information on how many hotels contain at least one of the sub-keywords of the main keyword, or the keyword itself. The final column shows the percentage of hotels that contain that keyword or sub-keyword.

Table 1: Keyword groups for the hotel attribute 'amenities'

Keyword	**Sub-items**	**Number of hotels**	**Percentage (%)**
internet	wifi	377	97.92%
parking		354	91.95%
non-smoking hotel		326	84.68%
bar	coffee shop, wine	300	77.92%
breakfast	complimentary	291	75.58%
laundry service	washing machine, ironing service, dry cleaning, clothes dryer	287	74.55%
restaurant	special diet menus, banquet room	272	70.65%
child	kid, baby	229	59.48%
summer activities	hiking, bicycle, golf course, horseback riding, tennis court, mini golf, table tennis, tennis court offsite, walking tours, squash, badminton, archery	222	57.66%
spa	massage, hot tub, hot spring bath, hammam, solarium, steam room, manicure, pedicure, waxing services, body wrap, foot bath, salon, facial treatments, sauna, makeup services	213	55.32%
airport transportation	shuttle bus service, car hire, taxi service	210	54.55%
meeting rooms	conference, business, executive lounge	209	54.29%
baggage storage		202	52.47%
pet friendly		191	49.61%

24-hour	concierge, butler, doorperson	174	45.19%
pool	beach	122	31.69%
gym	fitness, personal trainer, yoga classes	104	27.01%
skiing	ski	103	26.75%
terrace	outdoor furniture, sun loungers, umbrella, sun deck	99	25.71%
water activities	fishing, canoeing, water park, diving, snorkeling, boating, lazy river, windsurfing, water sport, waterslide	89	23.12%
shop	gift shop, vending machine, convenience store	88	22.86%
indoor entertainment	game room, board games, darts, billiards, bowling, newspaper, dvd, book	88	22.86%
evening activities	entertainment, casino, nightclub	42	10.91%
picnic	bbq, fireplace, outdoor dining	40	10.39%

3.2 Hostel keywords

The keywords for the ‘Hostels’ are the same as for the ‘Hotels’. This is because they were scraped together; they are grouped as one category in the TripAdvisor website, and then later we divided them into two groups. The keywords along with their sub-items and frequency of occurrence are presented in Table 2.

Table 2: Keyword groups for the hostel attribute ‘amenities’

Keyword	**Sub-items**	**Number of hostels**	**Percentage (%)**
internet	wifi	426	82.72%
parking		383	74.37%
non-smoking hotel		316	61.36%
summer activities	hiking, bicycle, golf course, horseback riding, tennis court, mini golf, table tennis, tennis court offsite, walking tours, squash, badminton, archery	238	46.21%
child	kid, baby	237	46.02%
bar	coffee shop, wine, happy hour	215	41.75%
restaurant	special diet menus, banquet room	209	40.58%
pet friendly		208	40.39%

laundry service	washing machine, ironing service, dry cleaning, clothes dryer	192	37.28%
airport transportation	shuttle bus service, car hire, taxi service	179	34.76%
breakfast	complimentary	160	31.07%
terrace	outdoor furniture, sun loungers, umbrella, sun deck, patio	134	26.02%
baggage storage		125	24.27%
spa	massage, hot tub, hot spring bath, hammam, solarium, steam room, manicure, pedicure, waxing services, body wrap, foot bath, salon, facial treatments, sauna, makeup services	110	21.36%
indoor entertainment	game room, board games, darts, billiards, bowling, karaoke, video game console, newspaper, dvd, book	106	20.58%
picnic	bbq, fireplace, outdoor dining	104	20.19%
meeting rooms	conference, business, executive lounge	103	20.00%
water activities	fishing, canoeing, water park, diving, snorkeling, boating, lazy river, windsurfing, water sport, waterslide	95	18.45%
skiing	ski	93	18.06%
shop	gift shop, vending machine, convenience store	80	15.53%
24-hour	concierge, butler, doorperson	73	14.17%
pool	beach	62	12.04%
gym	fitness, personal trainer, yoga classes	23	4.47%
evening activities	entertainment, casino, nightclub	22	4.27%

3.3 Restaurant keywords

In this group, keyword frequency was the most influential in selecting the main keywords. The reason is that almost all sub-items have a notably low frequency (between 0.1% and 2%). We can see in the table that even grouping similar cuisines leads to a low frequency in half of the keywords. This is to be expected, keywords such as ‘slovenian’, ‘european’ and mediterranean’ to have the highest frequency, much higher than cuisines from other continents.

One inaccuracy is using ‘irish’ as a sub-item for the ‘central european’ keyword. However, we used it here because it is not belong to any of the main keywords and has a frequency of

only 0.1%. All the keywords, their sub-items and frequency of their occurrence are shown in Table 3.

Table 3: Keyword groups for the restaurant attribute ‘cuisine’

Keyword	**Sub-items**	**Number of restaurants**	**Percentage (%)**
slovenian		583	61.05%
central european	austrian, french, czech, irish	269	28.17%
bar	cafe, wine, gastropub, pub, brew	246	25.76%
mediterranean	tunisian, african, spanish, greek, turkish, croatian	239	25.03%
italian	southern-italian, northern-italian, campania, lazio, neapolitan, romana	191	20.00%
pizza	neapolitan	167	17.49%
seafood		151	15.81%
grill	barbecue	122	12.77%
international		62	6.49%
eastern european	albanian, russian, ukranian, georgian	56	5.86%
asian	chinese, thai, japanese, indian, vietnamese, sushi, korean, bangladeshi, mongolian, nepali	48	5.03%
american	hawaiian, contemporary	45	4.71%
fast food		45	4.71%
steakhouse		25	2.62%
soups		25	2.62%
healthy		19	1.99%
south american	mexican, argentinean, brazilian, latin	17	1.78%
fusion		13	1.36%
middle eastern	lebanese, arabic, persian	12	1.26%
street food		12	1.26%

3.4 “Things to Do” keywords

The keywords for this category were chosen to include the highest percentage of items. This is the only category where we use keywords based on the most frequent words from user

comments. In TripAdvisor, this category did not have any useful information provided by the website, except the name, location and rating. That is why we chose to use this approach, otherwise we would have to recommend items only based on rating, with no context.
The variety of words and phrases scraped included 1600 unique keywords to describe the items. There are only 20 items with a frequency above 3%. That is why even our main keywords selected have a frequency lower than the keywords from the other categories. Nevertheless, all 1600 keywords are user searchable. These keywords are shown in Table 4 along with their sub-items and frequency of occurrence. This category would produce better recommendations using the knowledge-based recommender. It would offer a lower number of recommendations, but we expect that they would be close to exactly what the user is looking for.
The content-based recommender is more useful for everyday activities and ideas on how to spend your day. It would not work well if the user is looking for specific activities that he enjoys.

Table 4: Keyword groups for the “things to do” attribute ‘keywords’

Keyword	**Sub-items**	**Number of “things to do”**	**Percentage (%)**
museum		51	13.08%
park		50	12.82%
shop	store	37	9.49%
castle		35	8.97%
ski		33	8.46%
walking tour	walk	33	8.46%
town square	square	32	8.21%
nature		31	7.95%
winery	wine	30	7.69%
lake		29	7.44%
waterpark	water	29	7.44%
church		29	7.44%
bridge		22	5.64%
cave		14	3.59%

4 RECOMMENDER SYSTEM

As mentioned before each item has certain attributes. The initial values of each attribute are taken from TripAdvisor. We believe this is a better starting point than setting all the item's attributes to zero. As the users use the application and rate the items, their attribute values are adjusted accordingly. We calculate the average rating for the item attribute after adding the user's rating using the following formula:

$$Attribute_{rating} = \frac{(Attribute_{rating} * Number\ of\ Reviews) + User_{rating}}{Number\ of\ Reviews + 1} \quad (1)$$

This way the users can influence future recommendations and create a more accurate representation of each item for others. Through this, the system is able to create its own dataset, after being used for some time, which is different from the initial data collected from the TripAdvisor.

4.1 Knowledge-based recommender

After a user logs in for the first time we have no information regarding what he likes. That is why he has the option to set his preferences for the system to use as a starting point. Based on his choices the system can make recommendations. The first set of recommendations are made using only the knowledge-based recommender system. After he rates at least one item, the user model begins to change and the hybrid recommender can take effect. If he does not set any preferences, neither of the recommenders can be used. The user will get random recommendations until he either rates at least one item, or sets his preferences.

For the category of restaurants, the preferences include setting an overall minimal rating that a restaurant has to have to be recommended, a minimal rating for each of the attributes: 'Food', 'Value', 'Service' and 'Atmosphere'. Leaving any of them at zero, disregards that attribute when predicting recommendations. The maximum value is five. For the second part, he is given a list of keywords. The user can select as many as he wants. Initially all keywords have a value of zero. After he chooses his affinities, the value of those keywords is changed to a maximum value of five.

For the hotels and hostels categories, the preferences include setting the minimal acceptable rating for an item to have for 'Overall item rating', 'Value', 'Service', 'Location' and 'Cleanliness'. After the numbered attributes, similar to the restaurant preferences, he has a list of keywords (hotel amenities). He can select as many as he wants.

"Things to Do" only have the 'Overall item rating' as a numbered attribute, after that several keywords can be selected.

It is worth noting that the higher the rating preference and the more keywords selected, the less items will fit the criteria.
Besides using the knowledge-based recommender as a baseline for the content-based one, the system provides the functionality for strictly knowledge-based recommendations. The user can search for recommendations by using a search word or phrase and setting preferences, they can be different from those in the user model. Here, the user is not restricted to selecting pre-chosen keywords, but can input any keyword he chooses. In this type of recommendation, all items, which fit the user's criteria, are shown, not limiting his recommended items to 10 options.
The knowledge-based system takes the user's inputs as parameters. If an item is at least greater or equal than the rating specified by the user and contains at least one of the selected keywords it is shown as a recommendation.

4.2 User Model

We use the user model to predict items that the user would like. The user table in the database has the following columns:

- Name – set on account creation
- Email - set on account creation
- Birthday - set on account creation
- Username - set on account creation
- password (encrypted) - set on account creation
- hotels – vector of user hotel preferences
- rated_hotels – array of (hotel_id, user_rating) pairs
- hostels – vector of user hostel preferences
- rated_hostels – array of (hostel_id, user_rating) pairs
- restaurants – vector of user restaurant preferences
- rated_restaurants – array of (restaurant_id, user_rating) pairs
- things_to_do – vector of user "things to do" preferences
- rated_things_to_do – array of (thing_to_do_id, user_rating) pairs

The fields used to predict, for example restaurants, are 'restaurants' and 'rated_restaurants'. 'restaurants' is a string separated into 6 parts, each one of these is a number between 0 and 5:

- Minimal user acceptable overall restaurant rating
- Minimal user acceptable food rating
- Minimal user acceptable service rating

- Minimal user acceptable value rating
- Minimal user acceptable atmosphere rating
- Vector of values representing the keywords of restaurants

When the user rates an item, any rating above zero is taken into consideration. The attributes with numbered values, which are visible to the user, are then calculated using Equation 2:

$$User_{preference} = Item_{rating} + 0.5 * \left(Item_{rating} - User_{rating}\right) if\ User_{preference} = 0 \quad (2)$$

If Equation 2 results in a preference with value greater than five, the preference is set to 4.5. This is because setting the rating preference at 5 would significantly reduce the number of recommendable items. While if the equation result is below zero, the preference is set to 0.5. It is not set to zero because the user has rated an item and we have some information about his preference for that attribute.

$$User_{preference} = User_{preference} + 0.1 * \left(Item_{rating} - User_{rating}\right) if\ User_{preference} \neq 0 \quad (3)$$

Equation 3 is executed if current user preference is between 0.5 and 4.5, this ensures that there are enough recommendations that fulfil the conditions, and that the preference value is always between zero and five.

While Equation 4 calculates the value of the keywords:

$$keyword_{value} = keyword_{value} * 0.5 + keyword_{user\ rating} * 0.5\ if\ keyword_{value} = 0 \quad (4)$$

This equation is used when we have no information about the user preference for a given keyword. It drastically changes the rating from zero. After we have some information for the keyword, the weight of the new user rating is reduced to 0.1, as shown in Equation 5. This ensures small changes to the user model, we do not want to drastically change the recommendations with each new user rating given.

$$keyword_{value} = keyword_{value} * 0.9 + keyword_{user\ rating} * 0.1\ if\ keyword_{value} \neq 0 \quad (5)$$

The first time a user rates an item, it affects both the attributes of the item and the user model using the equations shown. After the first rating to an item, each rating a user gives to an already rated item, only affects the user model. This way one user cannot control how a restaurant is perceived through rating it multiple times. On the other hand, if his experience changes after multiple visits to the same place, he can rate it again to update his preferences in the user model. Items that receive an overall rating of 2.5 or below are no longer recommended to the user.

4.3 Content-based recommender

The content-based recommender starts working after user preferences have been set. Instead of comparing each item the user has rated with unrated items and finding recommendations similar to individual user-rated items, we create an 'ideal user item' using the selected preferences and the user ratings. Then, when searching for recommendations we use this 'ideal item' as a baseline for the comparison with the remaining unrated items. The adjustment in the user preferences is bigger when we have little or no information about the user, we can see this in Equation 4. Meanwhile, the adjustments to the user preferences are smaller when we have enough information in our user model to make adequate predictions, as seen in Equation 5.
The system tries to create the item, for each category, that would have all the characteristics important to the user. This way we have an idea what the user values in a given category. When searching for recommendations, it looks through all of the attributes of each item, compares them to the 'ideal user item', and only recommends those items which fulfill all of the requirements.
For the numeric attributes, items that are greater or equal than the user preference are taken into consideration. After this first selection, a second filtering process takes place. Only items, which contain one or more of the top five keywords (ordered by value) for the user, remain as viable recommendations. This remaining set is ordered by how many keywords from the user's top five each item contains in the end; the top 10 items are shown to the user. At the moment, this is the only order that can be used, in the future there could be a new functionality implemented offering a choice to the user: By which attribute would you like your recommendations to be sorted? This would increase the user's control over his experience, telling the system what he values most when asking for recommendations.

4.4 Example use case

A user logs in for the first time in the application. He is looking for some restaurant recommendations. The user sets his minimum preferences as follows:

- Rating: 3.5
- Food: 4
- Service: 4
- Value: 3
- Atmosphere: 0
- Preferred cuisines: Italian or Asian

He is currently in Ljubljana so he selects this region. The recommendations he receives are:

- Gostilnica 5-6 kg (4.42 rating, 466 reviews, 4.5 food rating, 4 service rating, 4 value rating, Southern-Italian cuisine)
- Kitajska restavracija Hong Kong (4.36 rating, 62 reviews, 4 food rating, 4.5 service rating, 4.5 value rating, 4.5 atmosphere rating, Asian cuisine)
- Pops Place Pizza (4.59 rating, 205 reviews, 4.5 food rating, 4 service rating, 4 value rating, Italian cuisine)
- Taj Mahal restaurant (4.36 rating, 247 reviews, 4.5 food rating, 4.5 service rating, 4 value rating, 4 atmosphere rating, Indian cuisine)
- Kitajski Vrt (3.97 rating, 41 reviews, 4.5 food rating, 4 service rating, 4.5 value rating, 4 atmosphere rating, Asian cuisine)
- Sushimama (4.28 rating, 303 reviews, 4.5 food rating, 4 service rating, 3.5 value rating, 4 atmosphere rating, Japanese cuisine)
- Maharaja (4.1 rating, 83 reviews, 4.5 food rating, 4.5 service rating, 4 value rating, 3.5 atmosphere rating, Indian cuisine)
- Maru Japosnak Restavracija Ljubljana (4.33 rating, 109 reviews, 4.5 food rating, 4.5 service rating, 4 value rating, 4.5 atmosphere rating, Japanese cuisine)
- Restavracija Mandala – vegan restaurant (4.97 rating, 30 reviews, 5 food rating, 5 service rating, Asian cuisine)
- Ogam (4.22 rating, 22 reviews, 4.5 food rating, 4.5 service rating, 4 value rating, Korean cuisine)

Since all of the restaurants offered have only Italian or Asian cuisine, none of them offers both, restaurants are not presented in any particular order. They are shown based on their id in the database.
For this example, we will assume that he visits the restaurant ‘Maharaja’. Afterwards he decides to rate his experience using the application: He gives the following ratings: overall rating 3, food 3.5, service 5, value 2, atmosphere 0 and the cuisine ‘Indian’, sub-item of ‘asian’ as 3.
These ratings would change the restaurant information to:

- Rating: 4.1 → 4.08
- Number of reviews: 83 → 84
- Food rating: 4.5 → 4.48
- Service rating: 4.5 → 4.51
- Value rating: 4→ 3.97
- Atmosphere rating: 3.5 → 3.5 (unchanged since user rating was 0)

The user model would be adjusted to:

- Rating preference: 3.5 → 3.61
- Food rating preference: 4 → 4.1
- Service rating preference: 4 → 3.95
- Value rating preference: 3 → 3.2
- Atmosphere rating preference: 0 → 0 (unchanged)
- Preferred cuisine: Italian 5 → 5, Asian 5 → 4.8

We can see that in cases where the user rating was lower than the restaurant rating, the user preference rating increases. This is because the system assumes that the user has a higher standard than the users that have rated the restaurant so far. Thus, adjusting his expectations accordingly.

5 IMPLEMENTATION

For the implementation of the application, we used Android Studio version 4.1. In this IDE, we used XML for frontend development, creating the Graphical User Interface (GUI). For the backend functionalities, we used Java SE 15. For a database, we used phpMyAdmin MySQL. In addition, PHP to create scripts for communication between the application and the database. Every version from Android 6.0 and above supports the application.

5.1 User Interface

The interface was created with 'ease of use' in mind. Showing only the information that helps the user to make a choice. It consists of several possible screens, including:

- Login / Signup (access for users / account creation for new users)
- User preferences (user can edit his preferences), each of the four categories has a different preferences screen
- Item recommendations (a list of recommendations for the chosen category)
- Item Selected (shows detailed description of an item)
- Rating (user can rate different parts of his experience)

Figure 1: List of recommendations

In Figure 1, we can see how the recommendations are shown to the user. Particularly here, we can see the recommended restaurants. Below the application name ('Trip planner') we can see which category we are currently seeing. Next to the category title, is the region

selector. It is implemented as a dropdown menu with the regions as selectable options. It shows the top 10 items for the selected region (it is possible for the user to receive a recommendation which is not top 10 for him in Slovenia). Below, are the actual recommendations. The user sees the top 10 recommended items based on his preferences. For each item, the most relevant information is shown at a glance: Name, Location and Cuisine.

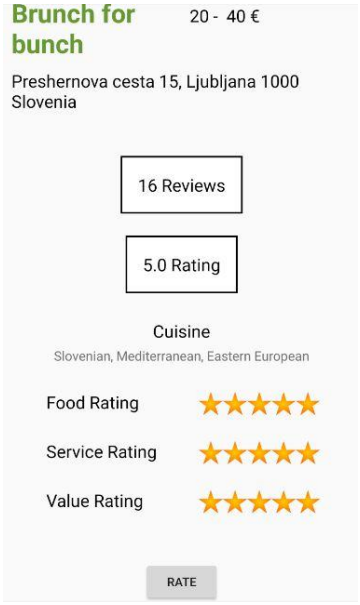

Figure 2: Recommended item details

After selecting an item, a more detailed view is opened, as shown in Figure 2.
We can see the name of the restaurant in the top left, with the specific location below the name. To the right is the price range for that restaurant (only 36% of restaurants have this attributes). Next, we can see the number of user reviews and the overall rating.
After that, the cuisines offered are shown as a comma-separated list. The last attributes are ‘Food’, ‘Service’, ‘Value’, and ‘Atmosphere’ (restaurant above is missing atmosphere rating). At the end is a rate button, this opens a dialog, as shown in Figure 3, where the user can rate different aspects of his experience.

Figure 3: Rating dialog

As shown in Figure 3, we can see the system offer the following aspects of a restaurant to rate:

- Overall item rating – rate the whole experience (required)
- Food rating – rating of food, could be overall food rating of the group (optional)
- Service rating – rating of the staff (optional)
- Value rating – rating of the dining experience compared to the money spent (optional)
- Atmosphere rating (optional)
- Cuisine rating – rating of the food, could be the same rating if the user was alone at the restaurant (optional); if cuisine is selected in the dropdown menu (required)

Next, we have the rating options for the ‘Hotels’ and ‘Hostels’ categories, the rating dialog is visually similar to Figure 3, except, some of the rating options are changed:

- Overall item rating – rate the whole experience, in these group categories the overall rating also changes the user’s preference rating for all of the amenities that the given hotel / hostel offers (required)
- Location Rating – rating of the hotel / hostel location (optional)

- Cleanliness rating – rating of the cleanliness of the room and the hotel / hostel overall (optional)
- Service rating – rating of the staff (optional)
- Value rating – rating of the experience compared to the money spent (optional)

Finally, we have the rating options for the "Things to Do" category. This category does not have different ratings for aspects of the user's experience. The rating dialog only offers two options:

- Overall item rating – rate the whole experience (required)
- Keyword rating – rate the keyword that describes the item (required)

5.2 Structure

5.2.1 Data-flow diagram

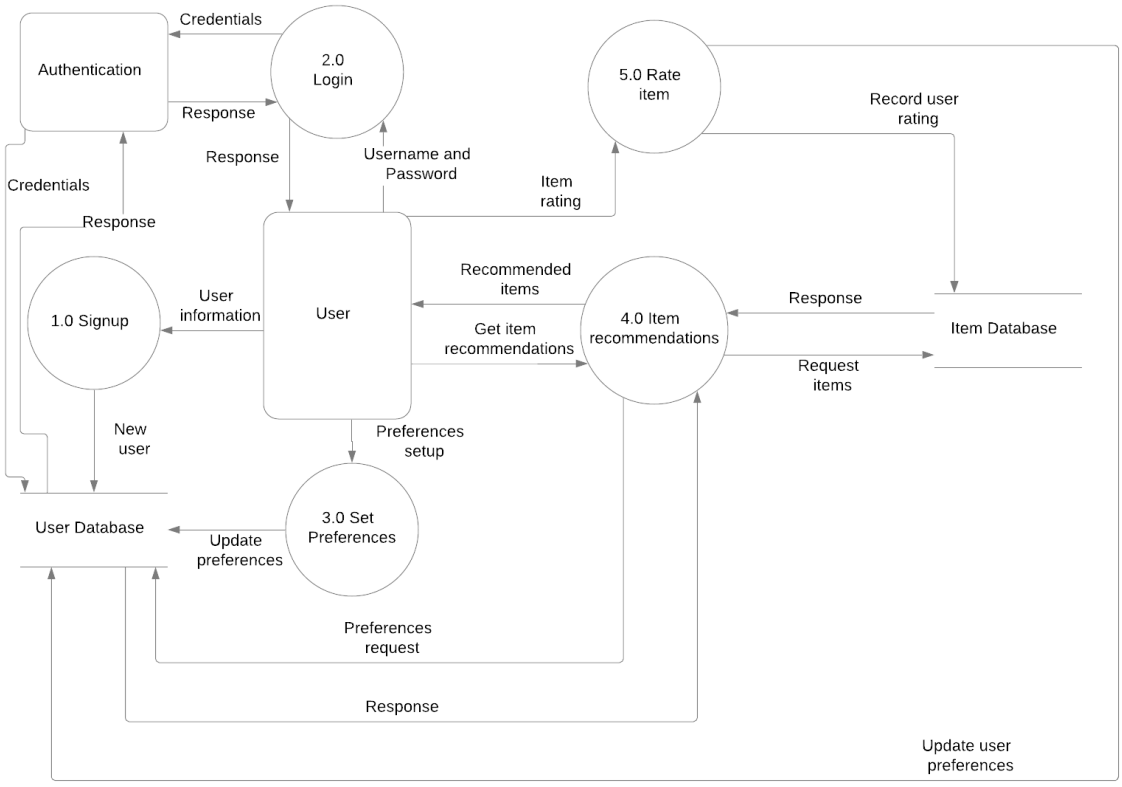

Figure 4: Level 1 data-flow diagram

The diagram in Figure 4 is used to explain how data is being processed and how it travels between the application and database based on the user's requests. In the diagram we can

see a more general flow, while below we will explain in detail what data passes through which activity screen.
To make the request we use the 'Volley' HTTP library available in Android Studio. Since we request data from the database whenever the user changes the active screen, we created a method that uses three parameters to execute this change and load the data that we received from the database into the new activity opened:

- Source activity: the current active activity
- Target activity: activity requested by the user
- Data to be fetched from the database: URL of the PHP script that retrieves the data from the database

The application consists of 16 activity screens in total:

- Sign up: requests no data from the database, sends user information to the database to be stored
- Log in: requests no data from the database, sends user credentials to the database for validation
- Hotels: requests hotel items from the database, displays list of 10 hotel recommendations for the current user
- Hostels: requests hostel items from the database, displays list of 10 hostel recommendations for the current user
- Restaurants: requests restaurant items from the database, displays list of 10 restaurant recommendations for the current user
- "Things to Do": requests "things to do" items from database, displays list of 10 "things to do" recommendations for the current user
- Hotel preferences: requests no data from the database, sends newly selected user hotel preferences to the database to be stored
- Hostel preferences: requests no data from the database, sends newly selected user hostel preferences to the database to be stored
- Restaurant preferences: requests no data from the database, sends newly selected user restaurant preferences to the database to be stored
- "Things to Do" preferences: requests no data from the database, sends newly selected user "things to do" preferences to the database to be stored
- Search: requests no data from the database, displays four buttons for the user, offering the options to search for hotels, hostels, restaurants or "things to do"
- Search hotels: requests no data from the database, uses user search parameters to request hotels that fit the user's criteria, redirects to 'Hotels' activity

- Search hostels: requests no data from the database, uses user search parameters to request hostels that fit the user's criteria, redirects to 'Hostels' activity
- Search restaurants: requests no data from the database, uses user search parameters to request restaurants that fit the user's criteria, redirects to 'Restaurants' activity
- Search "things to do": requests no data from the database, uses user search parameters to request "things to do" that fit the user's criteria, redirects to 'Things to Do' activity
- Rating dialog: requests no data from the database, sends user rating parameters to the database to be stored

5.2.2 Application structure diagram

Below, as shown in Figure 5, we display the structure of our application and how the user can move through the different activity screens. From the diagram, we can see that every session has to begin by logging in. After the user enters his credentials all of the other functionalities are available. First he is redirected to the 'Hotels' activity. After this, the user has full control of where he wants to go within the application. Figure 5 displays the diagram that shows how the activities are connected between each other.
Going from an upper to a lower row is done by interacting with the application (example: by pressing the "log in" button, you are automatically redirected to the hotel recommendations). In the second row, the user can switch between any two activities via a menu where all 5 activities are offered as options. While, going from a lower to a higher row is done by the back button that is already built into the android interface.

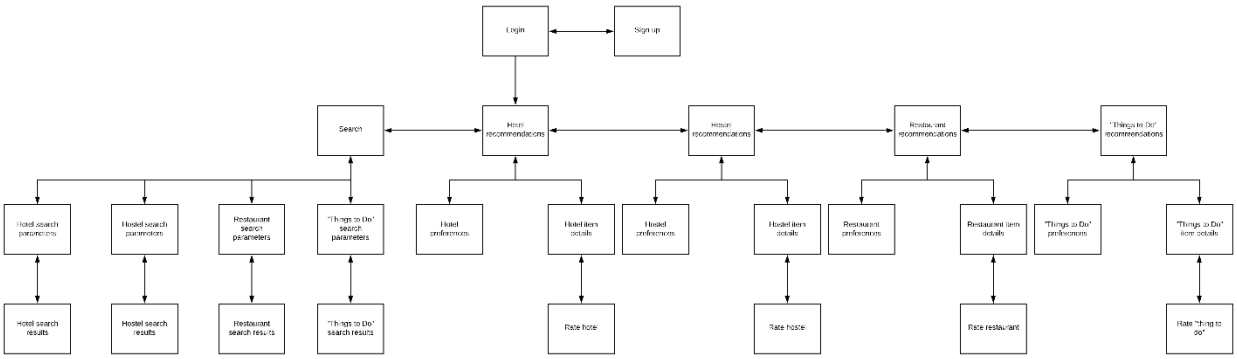

Figure 5: Application structure diagram

5.2.3 Entity-relation diagram

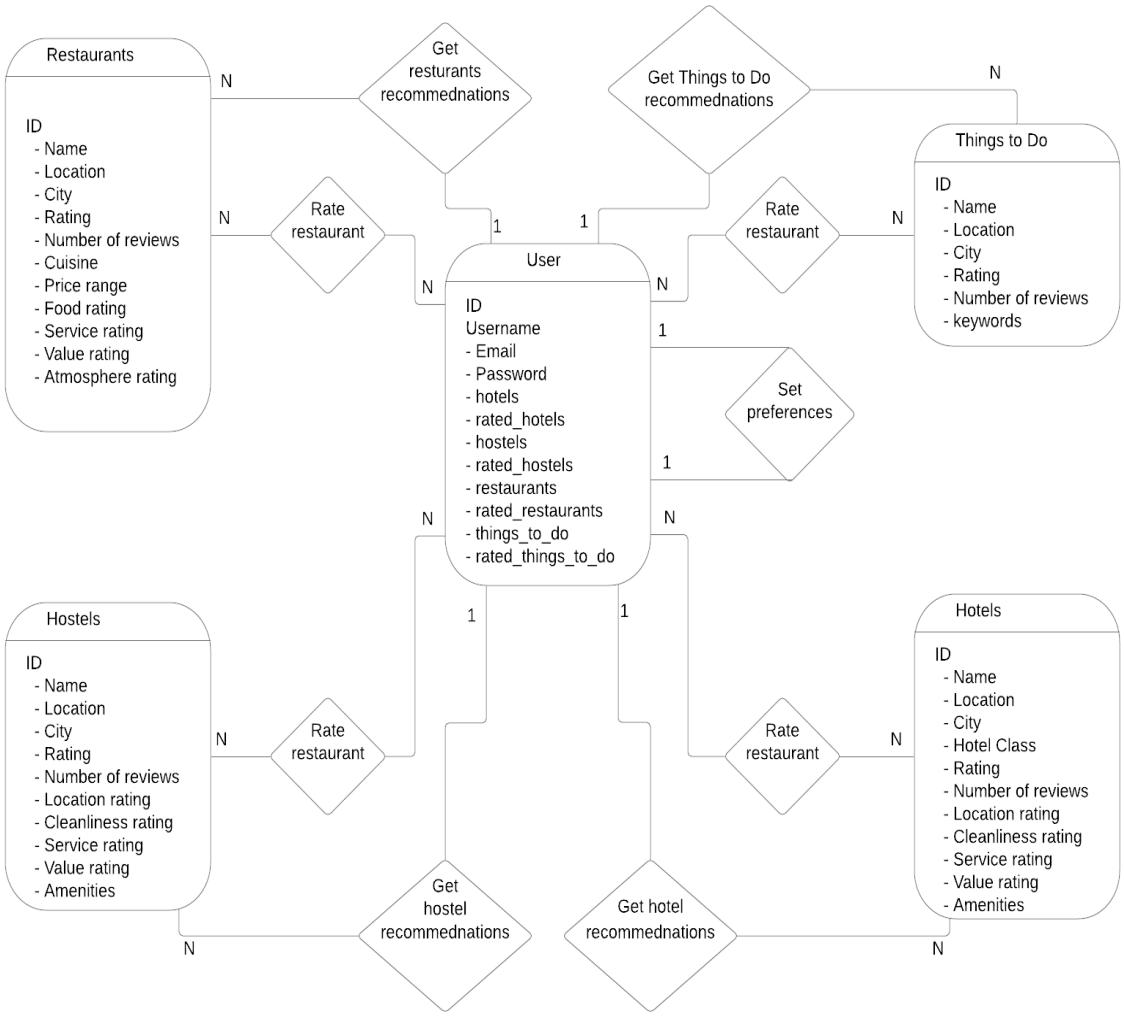

Figure 6: Entity-Relation diagram

In Figure 6, we show the database structure and how the individual tables are connected with each other. We have five tables in total. One for the user data and the others for each of the item categories.

The relations are between the user table and the four remaining tables. The user table has two similar relations (with small changes) with each remaining database. The first relation is when a user rates an item (hotel, hostel, restaurant, “thing to do”). The relation is “user N-N items”, meaning that one user can rate multiple items, and also one item can be rated by multiple users. The other relation is “get item recommendations”.

This relation is “user 1-N item recommendations”. This is because each user gets multiple item recommendations (up to 10 items) based on his preferences, but the list of recommendations is unique to each user based on the user’s preferences. Although it can happen that two users have the same preferences, it is unlikely.

The last relation is “user 1-1 preferences”, both attributes of the same table. This is because the user can edit his own preferences and each user can only have one set of preferences for each category.

6 CONCLUSION

The thesis presents the implementation of a mobile application to be used as a tour guide in Slovenia. We achieved this by using a recommender system. We decided on a knowledge-content hybrid recommender system. This type of system offers a good solution to predict items a user would like by using one system to cover the weaknesses of the other. On the other hand, using the knowledge part if the system to recommend items in rare cases when the user is looking for something completely different from his usual expectations. Situations where a group of people are making a decision, for example. This would allow the user to find different things from his everyday recommendations, while not affecting his individual preferences for further use in the future.

Based on this topic, future studies can be made regarding the equations used, and how it would affect the recommendations, if the numbers were changed. Alternatively, would it be better if a classic content-based recommender were implemented. In other words, would it be better if for each user session all of the rated items were compared to the item set to produce new recommendations.

Another approach would be to change some of the preferences used, not to use explicit user input, but to work in the background. To analyze how the user is interacting with the application. Swapping the content-based with a collaborative filtering system, or creating a hybrid between the two.

There are varieties of ways this can be changed or built upon, but we believe it is a good starting point to build a useful tool that offers users the ability to discover places throughout Slovenia.

7 DALJŠI POVZETEK V SLOVENSKEM JEZIKU

V okviru diplomskega dela smo izdelali mobilno aplikacijo, ki ima vgrajen priporočilni sistem za priporočanje hotelov, hostlov, restavracij in aktivnosti po vsej Sloveniji. Aplikacija deluje na operacijskem sistemu Android.

Za izgradnjo priporočilnega sistema smo potrebovali začetne podatke na katerih temelji naš priporočilni sistem in nenazadnje tudi aplikacija. Odločili smo se, da za ta namem uporabimo javno objavljene podatke na portalu TripAdvisor. Izbor podatkov na portalu smo izvedli tako, da smo iskanje omejili z uporabo ključne besede ‘Slovenija’ in nato zbrali podatke iz vnaprej določenih zavihkov na spletnem mestu z oznako ‘Hoteli’, ‘Restavracije’ in ‘Aktivnosti’. Prenos podatkov smo opravili s pomočjo namensko izdelane skripte. Ko smo imeli vse potrebne podatke, smo se lotili čiščenja in prilagoditve le-teh. Najprej smo odstranili vse primere, ki niso bili locirani v Sloveniji a jih je TripAdvisor vseeno vrnil kot rezultat iskanja. Nato smo dodali dodaten atribut, imenovan 'Regija', katerega vrednost je bila določena na podlagi poštne številke naslova primera. Možne vrednosti tega atributa so dejanske regije na katere je razdeljena Slovenija. Primere, ki pripadajo skupini 'Hoteli' smo razdelili na hotele in hostle. Tisti primeri, ki so imeli podatek o hotelski kategoriji, so ostali v skupini 'Hoteli', vsi tisti primeri, ki kategorije niso imeli določene, so bili prestavljeni v skupino 'Hostli'.

Vsaka od teh skupin ima en atribut, ki vsebuje vrsto ključnih besed, ki opisujejo najpomembnejše lastnosti primera. Pri hotelih in hostlih gre za atribut udobja, pri restavracijah gre za vrsto kuhinje, pri aktivnostih pa so to najpogostejše besede, ki se pojavijo v komentarjih uporabnikov na portalu TripAdvisor. Pregledali smo vse besede in izmed njih izbrali približno 15-25 najustreznejših ključnih besed za vsako kategorijo in jih uporabili kot glavno ključno besedo. Za najustresnejše ključne besede smo izbrali tiste, ki pokrijejo oz. pomensko združujejo čim več drugih ključnih besed. Ključne besede, ki niso bile izbrane kot glavne ključne besede, so tako postale podpomenke posameznih ključnih besed.

Ko se uporabnik prijavi v aplikacijo, lahko nastavi svoje preference za vsako skupino. Se pravi, da določi minimalno sprejemljivo oceno za ocenjene attribute in izbere ključne besede, ki so mu najbolj pomembne. Tako se ustvari začetni “popoln” model uporabnika, ki ga uporabimo za primerjavo z naborom primerov, ki jih imamo v naši bazi. Vsak primer, ki ustreza vsem minimalnim zahtevam, je primeren za vključitev v naše priporočilo. Takšne primere nato razvrstimo glede na to, koliko uporabnikovih izbranih ključnih besed vsebujejo. Uporabniku priporočimo 10 primerov, ki so razvrščeni najvišje. Del modela uporabnika se uporablja kot izhodišče za vsebniska priporočila.

Poleg tega načina podajanja priporočil lahko priporočamo tudi na podlagi znanja. Uporabnik lahko izbere merila za drugačen nabor priporočenih primerov, kar je koristno, ko išče kaj drugačnega od običajnih pričakovanj.

Vrednosti atributov v modela uporabnika in primera so zasnovani tako, da se spreminjajo z vsako oceno uporabnika. Vsaka ocena uporabnika ima vpliv na vrednost atributov primera. Tako se bodo ocene atributov primerov skozi uporabo sistema spremenile, da bomo imeli bolj natančne ocene atributov posameznega primera. Vrednost atributa se spremeni z uporabo enačbe:

$$(ocena_{element} * število\ ocen + ocena_{uporabnik})/(število\ ocen + 1)$$

Model uporabnika se ustvari po tem, ko uporabnik nastavi svoje minimalne preference ali oceni vsaj en primer. Za vsako kategorijo so nastavitve shranjene kot vektor vrednosti. Vsaka vrednost je število med nič in pet. Prvih pet elementov vektorja je najmanjša sprejemljiva ocena, ki jo mora imeti primer, da se šteje kot dobra napoved za uporabnika. Zadnji element vektorja je polje z vrednostmi, ki predstavljajo preference uporabnikov za vsako glavno ključno besedo za to kategorijo.
Ko uporabnik oceni primer, se te nastavitve spremenijo glede na oceno primera, oceno, ki jo je uporabnik dal, in katere ključne besede je izbral. Pri prilagajanju modela uporabnika se upoštevajo le ocene, ki niso enake nič, spremeni pa se le vrednost za izbrane ključne besede. Menimo, da je predstavljeni hibridni sistem priporočil dober pristop k priporočanju izdelkov uporabnikom. Model uporabnika in izbira začetnih minimalnih preferenc zmanjšuje problem hladnega zagona sistema in omogoča boljša priporočila že od samega začetka. Sistem se v prihodnje lahko nadgradi tako, da se ga dopolni še z metodo skupinskega filtriranja, ki bi omogočala še boljša priporočila in prilagajanje modela tudi glede na ocene primerov drugih uporabnikov.

8 REFERENCES

[1] C. C. Aggarwal. *Recommender systems*. Vol. 1. Cham: Springer International Publishing, 2016.

[2] P. Brusilovsky and E. Millán. "User models for adaptive hypermedia and adaptive educational systems." *The adaptive web*. Springer, Berlin, Heidelberg, 2007. 3-53.

[3] R. Burke. "Integrating knowledge-based and collaborative-filtering recommender systems." *Proceedings of the Workshop on AI and Electronic Commerce*. 1999.

[4] E. Çano and M. Morisio. Hybrid recommender systems: A systematic literature review. Intelligent Data Analysis 21(6) (2017): 1487-1524.

[5] N. Hariri, et al. "Context-aware recommendation based on review mining." *ITWP@ IJCAI*. 2011.

[6] MH. Park, JH Hong, and SB Cho. "Location-based recommendation system using Bayesian user's preference model in mobile devices." *International conference on ubiquitous intelligence and computing*. Springer, Berlin, Heidelberg, 2007.

[7] F. Ricci, L. Rokach, and B. Shapira. "Recommender systems: introduction and challenges." *Recommender systems handbook*. Springer, Boston, MA, 2015. 1-34.

[8] K. Zhang, et al. "Hotel recommendation based on user preference analysis." *2015 31st IEEE International Conference on Data Engineering Workshops*. IEEE, 2015.