RADBOUD UNIVERSITY NIJMEGEN

MASTER THESIS

Composite recommendation and persona-ization in a shopping environment

Author:
Senna VAN IERSEL

Supervisors:
Prof. dr. ir. Arjen DE VRIES
Dr. Jason FARQUHAR

Artificial Intelligence - Web and Language Interaction
Faculty of Social Sciences

March 22, 2017

RADBOUD UNIVERSITY NIJMEGEN

Abstract

Faculty of Social Sciences
Artificial Intelligence - Web and Language Interaction

Composite recommendation and persona-ization in a shopping environment

by Senna VAN IERSEL

A mobile and web application are implemented to test *composite* recommendation in a shopping environment. Routes along shops in the center of Nijmegen are recommended and a comparison is made with recommendations by a *content-based* recommender system. The *composite* recommender system makes diverse sets of shops in a route, whereas the *content-based* system recommends the top-n shops of a user. This research is thus a start in finding what type of recommendations is preferred and whether the diversity of the *composite* routes is important in a shopping environment. Because we have no prior information about our participants, we encounter the *cold start* problem. The appliance of *persona-ization* as a method to address this problem is tested. *Persona-ization* is a method in search problems to search on behalf of others. In this study, we apply it to the recommendation problem where the first recommendation to each user is recommended on behalf of a self chosen persona that fits the user best.

A list of shops in the center of Nijmegen is constructed with each shop having one or more tags. These tags give information about the type of shop and are used to compute ratings of users for each shop. The same tags are used to describe the different personas. Preferred tags have a positive weight and tags that a persona dislikes have a negative weight. A user starts with the same weights as the self chosen persona.The user's feedback on what shops in the route are preferred is used to update the weights to improve future recommendations.

An experiment is set up that consists of two parts. In total, 39 participants completed the first part where the applicance of *persona-ization* was tested. Of those 39 participants, 27 also completed the second part where the performance of the *composite* recommender system is compared to the performance of the *content-based* recommender system. Results show that *persona-ization* significantly improves the recommendations and thus it is concluded that it addresses the *cold start* problem effectively. The *content-based* recommender system outperforms the *composite* recommender system; it is not only preferred significantly more often, the average percentage of shops that is indicated as personalised is significantly higher for the *content-based* recommended routes.

Contents

Chapter 1

Introduction

We all have experienced this scenario: we step inside a store, look for interesting products and an employee of the store walks towards you, asking whether you need some help with deciding what you want to buy. The employee tries to help you find a product that suits your personal needs. The profession of these employees is called *personal shopper*. A *personal shopper* thus helps finding products by giving personalized advice and recommendations. In the current study, we aim to perform a task similar to that of a personal shopper using our software PERSONAL SHOPPER (PESH), implemented as webapp and as mobile Android application. Instead of recommending products, the aim of PESH is to recommend a route along stores personalized to the users' tastes. All software and code together with results from the experiment can be found on https://github.com/Sennaa/PeSh.

1.1 Motivation

In this project two aspects are researched. Firstly, the application of *recommender systems*, systems that try to make personalized recommendations. In this research we focus on two types of recommender systems, *composite recommender systems* and *content-based recommender systems*. The first type tries to recommend a composite set of items that are complementary. The other type aims to make personalized recommendations based on the content of the set of possible recommended items. We will try to apply *composite recommendation* in a shopping environment and measure its performance with respect to a *content-based recommendation system*. The second aspect of this project is the application of *persona-ization* as a method to address the *cold start* problem and the perceived influence on recommendations in terms of how personal they are. The *cold start* problem is a common problem in recommender systems that arises when recommendations are made for new users. Because initially there is no information about a new user, the recommender system cannot immediately give personalized recommendations. We aim to reduce the problem of the *cold start* by applying *persona-ization*, a method where information about a user is gathered by attaching a fictional character or *persona* to the user that describes a type of user.

The first aspect is interesting because potentially improved recommendations can be made in a shopping environment. The list of shops obtained using *composite recommendation* is possibly more complementary, cohesive and relevant than the list obtained by a *content-based recommender system*. This can lead to users perceiving the *composite recommendations* as more favorable in comparison with the *content-based recommendations*. The aspect of *persona-ization* is of interest because it can contribute to a more personal perception of recommendations. Further, it can help overcome the *cold start* problem, by classifying a user to a certain persona. *Persona-ization* is also a way to handle the lack of data that the app has in its starting phase. Having few users and little feedback from these users makes the process of recommending items based on the behavior of similar users less accurate, because there is less data that can be used for these recommendations. With *persona-ization*, a "fake user" with certain preferences is created. The user selects the persona he identifies with the most so that extra information about the user is gained. With this information, the lack of data can be handled.

In addition to a potentially more personal perception of recommendations and a method to potentially overcome the *cold start* problem, the application can help users to find stores that fit to their needs. That way, it is shown how the research can be used in practice. We will apply *composite recommendation* in a shopping environment. This type of recommendation has already been applied to recommend points of interest (POIs) (Xie, Lakshmanan, and Wood, 2011). Not only a different environment is used, we also use a different approach in the current study, making use of the algorithms described for composite retrieval (Amer-Yahia et al., 2014). Therefore, it is interesting to research the application of *composite recommendations* in the shopping environment.

In summary, the impact of the current study can be a better insight in composite recommendation, especially in the shopping environment. A new method to address the *cold start* problem might be found that potentially will make it easier to give personal recommendations to users. Results of this study can give information whether users prefer to visit a more complementary set of shops or that complementarity is of less importance. Whereas in *composite retrieval* multiple bundles can be recommended, only one bundle of shops is recommended in this study, namely the route. Since there is only one bundle, there can be no interbundle diversity in the recommendation, therefore this type of diversity does not play a role.

1.2 Research Questions

The current study aims to apply *composite recommendation* to a shopping environment. The study investigates whether *composite recommendations* of stores are appreciated more than recommendations of stores by a *content-based recommender system* that makes its recommendations based on item similarity. Both systems recommend a route along a set of shops.

The evaluation of the recommender systems considers a combination of two criteria:

1. *Quality*: how does the user perceive the quality of the route, e.g. how well do the shops reflect the user's preferences?

2. *Time budget*: how well did the recommended route fit within the user's time budget?

These criteria will be discussed in more detail in section 6, that focuses on the evaluation of the research.

Thus, to get a better insight in *composite recommendation* in the shopping environment, the current study aims to answer the following research questions:

1. (a) Can *composite recommendation* be applied to a shopping environment?
   (b) Does *composite recommendation* outperform a *content-based recommender system* in a shopping environment, based on the above mentioned criteria?

The current study applies *persona-ization* as a technique to address the *cold start* problem. This will be attempted for both the *composite recommender system* and the *content-based recommender system*. The study also tries to find whether recommendations by both recommender systems are being perceived as more personal when *persona-ization* is used to address the *cold start* problem. Thus, we also aim to answer the following research questions.

2. (a) Can *persona-ization* be applied as a method to address the *cold start* problem in both *composite recommendation* and a *content-based recommender system*?
   (b) Are recommendations by both *composite recommendation* and a *content-based recommender system* perceived as more personal when applying *persona-ization* to address the *cold start* problem?

1.3 Outline

The outline of this thesis is as follows. In chapter 2, related research will be discussed about composite retrieval, recommender systems and the combination of both. Also, we discuss the background of persona-ization. The collection of data, assignment of tags to stores and the composition of personas are the topic of chapter 3. The actual implementation of both an Android- and web application, together with the integration of the recommender systems and persona-ization will be discussed in chapter 4. The experiment that is run to answer our research questions is explained in chapter 5 and its evaluation can be found in chapter 6. The results that are found are shown per research question in chapter 7. We end with a conclusion, discussion and possible future work, based on the results, in chapter 8.

Chapter 2

Related work

To perform the task of personal shopper, a personalized advice and recommendations of stores must be given to the users of PESH, making a personalized route recommendation. In the current study two types of recommendations are compared. Recommendations of a *content-based recommender system* are compared to more diverse recommendations of a *composite recommender system*. The latter type of recommender system is a modification of the problem of *composite retrieval*, where the field of use is altered from search problems to recommendation problems. However, the approach used in *composite retrieval* to retrieve items is the same as the approach used in the *composite recommender system* to make recommendations.

When little information is known about a user, the task of making a personalized recommendation is hard. To make this task less hard, this study applies the concept of *persona-ization*. With *persona-ization* the *cold start* problem can be handled. Further, it helps overcome the problem of having no prior information about a user because the user's preferences are obtained.

2.1 Composite Retrieval

Composite retrieval (Amer-Yahia et al., 2014) is the problem where k bundles of complementary items must be found. Such a bundle should contain items that together fit to a common purpose. For example, when you go camping, you need several equipment such as a tent, a sleeping bag, an accommodation and transport. Composite retrieval aims to fit items related to different aspects of the user's needs in a bundle. Items within a bundle are preferably relevant to these needs as well as diverse and cohesive (Bota, Zhou, and Jose, 2015, Bota et al., 2014).

The problem of composite retrieval is NP-hard (Amer-Yahia et al., 2014). It is intuitive that for the camping example there are many possible bundles of equipment. What type of tent fits best to a specific sleeping bag? It is hard to find the bundle that contains the most diverse, cohesive and relevant equipment. Moreover, there can be additional constraints such as a maximum budget that need to be taken into account. Because the problem of composite retrieval is NP-hard, different approximation algorithms have been proposed. These can be divided into *Produce-and-Choose*, *Cluster-and-Pick* and *Integer Programming*, where the latter is used as benchmark. An aspect that needs to be taken into account when choosing an approach is running time, which can differ for the two approaches depending on how large the problem instances are.

2.1.1 Produce-and-Choose

In the *Produce-and-Choose* approach, first several bundles are produced and a selection of these bundles is chosen. Two options are considered for the *production* phase. The first option is *Constrained Hierarchical Agglomerative Clustering (C-HAC)* (Davidson and Ravi, 2005). Here, each input element starts as a single cluster, after which iteratively the closest clusters are merged until a constraint is met. The other option is *Bundles One-By-One (BOBO)* (Amer-Yahia et al., 2014), based on *k-nn* clustering. In this method the input elements are iteratively chosen as pivot, around which a bundle is created. Good bundles are kept, others are removed. For the *choose* phase, a graph is built with bundles being the nodes and distances between bundles being the edges. A chosen set of k bundles is the subgraph with the maximum sum of edge weights, consisting of k nodes. The *Produce-and-Choose* approach is especially useful when diversity is not highly important.

2.1.2 Cluster-and-Pick

The *Cluster-and-Pick* approach, where the first step is to make clusters of items after which a valid bundle is picked from these clusters, is more useful when diversity is important. The clustering can be done using a standard clustering algorithm. In (Amer-Yahia et al., 2014) the clustering algorithm MeTiS (Karypis and Kumar, 1995) is used to create clusters having similar items within a cluster and dissimilar items in different clusters. Then, the best bundle is picked by iterativily checking the score of the bundles.

2.2 Recommender Systems

Preferences of users can be predicted by recommender systems (Resnick and Varian, 1997) in order to make personalized recommendations. Recommender systems are often divided in three categories (Adomavicius and Tuzhilin, 2005). Firstly *content-based recommender systems*, where recommendations are made based on prior items that were preferred by the user. Secondly, *collaborative recommender systems*, where recommendations are made based on what items similar users prefer. The third category consists of the *hybrid recommender systems*. Here, *content-based recommender systems* and *collaborative recommender systems* are combined. That way, recommendations are made based on the similarity between the items and on the users' behavior.

Another way of grouping types of recommender systems is the division in context-aware recommender systems (CARS), that take the context of a user into account, and recommender systems that do not take the user's context into account. Context-aware recommendations can be more relevant and personal (Adomavicius and Tuzhilin, 2005). CARS have been used to build mobile tourist applications (Setten, Pokraev, and Koolwaaij, 2004, Poslad et al., 2001). These systems make use of the context of users, e.g. the local time, location and agenda. The advantage of this is that the users can receive recommendations fitting in their schedule and location. A context-aware recommender system using the user's location as context has previously been applied in a similar environment as we use in this project, namely a shopping environment (Yang, Cheng, and Dia, 2008). Here, the interest of a user in a vendor's webpage is calculated taking into account the actual distance of the user to the vendor as well as the similarity between the user and the webpage.

A common problem in recommender systems is the so-called cold start problem or new user problem. Often, there is no information about new users which makes it hard to make personalized recommendations. Several methods have been proposed to deal with this problem (Rashid et al., 2002). An example of a method that uses information that is not user-specific is recommending popular items. Another method that is used is to recommend items with a high entropy, meaning that users have very divergent opinions about these items. Feedback of a new user on such items is worth more because the recommender system learns more about the user than when it gets a user's feedback on a popular item, because with the popular item the chance is higher that the new user will like it as well. In other methods user-specific information is obtained to improve personalized recommendations, such as personality information (Tkalcic et al., 2011).

2.3 Composite Recommendation

The idea to bundle results has been used most prominently in the domain of search, referred to as *composite retrieval*, however the composite notion has been applied to recommender systems as well (Xie, Lakshmanan, and Wood, 2011), as composite recommendation. In composite recommendation, one or more bundles of complementary items are recommended. In (Xie, Lakshmanan, and Wood, 2011), a mobile application was made for travel planning using composite recommendation. Here, k bundles of recommendations were made for different points of interest (POIs). Users could indicate a budget for both money and time. These budgets were taken into account in the application to better fit the needs of the user. For example, it was assumed that POIs that have a larger surface are on average being visited for a longer time. Also, the time to go from one POI to another was taken into account.

2.4 Personalized route recommendation

PESH aims to make a personalized route recommendation along stores. The mobile application described in the research above (Xie, Lakshmanan, and Wood, 2011) is an example of a personalized route recommender system, with a recommended route along POIs. In (Anagnostopoulos et al., 2016), a tour recommender system for groups is implemented. The aim is to find the best tour the group could perform together. Here, a selection of POIs is also made along which a route is recommended. Several algorithms are tested of which the Ant-Colony-Optimization algorithm is preferred due to its high performance in the recommendations and because it is significantly faster than other tested algorithms with a similar performance.

2.5 Persona-ization

A persona is a fictional character that describes a type of user in the target group (Pruitt and Grudin, 2003). It is often used in user-centered design and marketing. Recently, it has been proposed to use personas in search-related problems. Here, the persona is another (fictional or non-fictional) person on whose behalf a query is searched. The process of fitting results to such a query is defined as persona-ization (Bennett and Kiciman, 2015). To clarify this process, imagine that a mother is searching for a gift for her 20-year old son. In a search engine she writes the query "gifts for 20 year old boy". Assuming the search engine knows the demographics of the mother (say, 50 year old female), it can be extracted from this query that she is searching on behalf of someone else, namely a 20 year old male. Using previous search data from 20 year old males, an improved result to the query can be given. The process of persona-ization can even be extended when extra information is added in the query such as preferences of the person on whose behalf is searched.

Chapter 3

Data collection

Recommendation of routes along stores is based on a selection of stores with one or more tags. The tags are keywords that describe the type of the store it is assigned to. Section 3.1 discusses how the stores and tags were selected and how the tags were assigned to the stores.

Additionally, a composition of personas is required. Users are assigned to a persona that fits them best, therefore divergent personas are necessary. Section 3.2 describes how the personas are created.

3.1 Store-Tags assignment

A selection of stores in the city Nijmegen is used in PESH. Only stores in the town center of Nijmegen are used. Every store has between one and nine tags assigned, with a mean of 2.259 and a standard deviation of 1.144.

3.1.1 Store selection

The selection of stores consists of a subselection of the stores that can be found on http://www.centrumnijmegen.nl/winkels. The subselection of these stores consist of the stores that sell products. Stores that only offer services are left out, assuming that users are looking for products when shopping. Altogether, the types of stores that are left out are listed in table 3.1.

Further, the name of the store *"Het Huis de Brillenmaker Opticiens"* is changed to *"Eye Wish Plein 1944"* since the last has replaced the first. The stores that are left after these adaptations, are selected to be used in PESH.

Of all shops, a few features were used in the mobile app. They are listed below.

1. *Name* Taken from http://www.centrumnijmegen.nl/winkels.
2. *Address* Street name and house number.
3. *Estimated Spending Time* The estimated time a user would spend in the shop. The time is in minutes and the value is set to 10 minutes for each shop. Unlike in (Xie, Lakshmanan, and Wood, 2011), where POIs are assigned an estimated spending time proportional to the size of the POI, we use an estimated value that is equal for each shop. Shops are a subset of all POIs in an area where intuitively one can spend as much time in a smaller shop as in a larger shop. For example, some people spend much time in a small record store looking for an LP record or in a small book shop looking for specific books. Without additional

TABLE 3.1: Deleted types of stores

Hairdresser's salons
Beauty salons
Gold purchase companies
Tattoo shops
Publishers
Real estate companies
Lawyers
Escape Rooms
Repair shops
Designers (e.g. interior designers)
Tanning salons
Practices (e.g. homeopathy, massage, etc.)
Event organizations
Entrepeneurial shops
Travel agencies
Photo studios
Stores outside of the center of Nijmegen
Stores that can only be visited by appointment
Stores that are permanently closed

information about shopper behaviour, we use an equal value for the estimated time spend in each shop.

4. *Tags* See section 3.1.2. Per shop *s*, the tags are represented as a binary list where a *zero* at index *i* indicates that the tag at index *i* is not assigned to shop *s* and a *one* at index *i* indicates that the tag at index *i* is assigned to shop *s*.

5. *Similarities* The *shop similarities* $S(s_1, s_2)$ between two shops s_1 and s_2 are calculated as

$$S(s_1, s_2) = \frac{\sum_{tag \in tags} tag(s_1) == 1 \; and \; tag(s_2) == 1}{size(tags)}$$

where $tags$ is the binary list that indicates whether a tag is assigned to the shop and $size(tags)$ is the length of that list. $tag(s)$ is the binary value of the tag in $tags$ for shop s.

6. *Coordinates* The latitude and longitude of each shop is retrieved using Google Maps Distance Matrix API.

3.1.2 Tag selection

On http://www.centrumnijmegen.nl/winkels, each store is assigned to one or more categories, ranging from *Dameskleding (ladieswear)* to *Computerwinkel (computer store)*. We use these categories as tags in our app. On https://www.wugly.nl/plaatsen/254/nijmegen/ the stores in Nijmegen are also categorized. Each store that is in our selection of stores and can be found on this website is extended with the assigned categories as tags. This is only done when the assigned category gives extra information to the already selected tags, meaning that only tags that are synonyms to already existing tags in our selection are not added to this selection. This means the following in practice:

- The tag *Telecom* is not added to the selected tags since the list contains the tag *Telefoonwinkel (phone shop)* and *Telecom* is considered to be a synonym of *Telefoonwinkel*.
- The tags *Cadeau (gift)* and *Speelgoed (toys)* are added separately because they give more specific information than the already selected tag *Kado & Speelgoed (gift and toys)*.
- The same holds for *Optiek (optics)* and *Juwelier & Horloge (jeweler & watch)*. These tags are added because they give more specific information than the already selected tag *Juwelier & Optiek (jeweler & optics)*.
- *Mode accessoires (fashion accessories)* is added because it is assumed to be more specific than *Accessoires (accessories)*.

Some adaptations were made to the assigned tags to the stores. This was only done when it was obvious that the assigned tag was a mistake (deletion or modification of a tag) or it was well-known that a certain tag should belong to the store (addition of a tag). These adaptations were the following:

- The tag *Schoenen (shoes)* was assigned to the store *Brillencentrale Francissen Optiek B.V.*. The tag is changed to *Optiek (optics)*.
- The tag *Vloer, Wand & Raam (floor, wall & window)* is deleted from the store *Vinylarchief*.

The store-tags combinations that are obtained after above described modifications are saved and the address of all stores is added to be able to determine a route along the stores. The full list of tags can be found in Appendix A.

3.2 Persona composition

Personas can be considered as prototype users (Pruitt and Grudin, 2003). Because of the possible variety of users, the personas should be diverse. Further, they should make good use of the existing selection of tags. Since the tags are assigned to the stores, using the tags for the personas makes it easier to 'recommend' a store to a persona. Therefore, a list of personas is created that contains the name, a list of positive tags and a list of negative tags per persona. The positive tags describe the categories that are preferred by the persona and the negative tags describe the categories that are negatively associated with the persona. The same tags are used for the personas as for the description of the stores. An attempt is made to make a diverse list of personas. In order to do this, the personas are chosen based on departments in some large e-commerce companies (https://www.amazon.com, https://www.bol.com, https://www.aliexpress.com). The list of positive tags and the list of negative tags per persona can be found in Appendix B.

3.2.1 Persona assignment

Every user is assigned one persona. This assigned persona is chosen by the user himself. In the application, a list of all personas is shown from which the user should select the persona that resembles his preferences most. The list that is shown to the users consists of all the names and either the descriptions of the personas (mobile application) or the positive and negative tags (webapp).

3.2.2 Rating composition

The tags belonging to a persona are converted to ratings for all shops. For each shop s, the rating $R(p_s)$ of persona p for shop s should be high when many of the positive tags of p are assigned to shop s. The rating $R(p_s)$ of persona p for shop s should be low when many of the negative tags of p are assigned to s. Taking this into account, ratings are calculated based on the positive and negative tags as

$$R(p_s) = \frac{\#pos_{p\&s} - \#neg_{p\&s}}{len(tags_s)}$$

where $\#pos_{p\&s}$ is the amount of tags that are both assigned to the shop and as a positive tag to persona p and $\#neg_{p\&s}$ is the amount of tags that are both assigned to the shop and as a negative tag to persona p. $len(tags_s)$ is the total amount of tags that shop s has. For example, if only the tag *Elektronica (electronics)* is assigned to shop s, $len(tags_s) = 1$.

Chapter 4

Implementation

PESH is an application implemented in two different platforms. Firstly, it is implemented as an Android application for mobile Android devices. Secondly, it is implemented as a web application. In this chapter, the setup of both implementations is discussed in more detail. Also, a walkthrough through the application on both platforms is given as well as the lifecycle and database management per platform. Further, information about the contents of the database and about the implementation of persona-ization and the recommender systems is given.

4.1 Application set-up

4.1.1 Android application

PESH is implemented as an Android application, written in Java using Java SE Development Kit (JDK) 8. It is developed using Android Studio 2.1.2. The back-end and front-end of the application are separated, where an *activity* is a front-end screen. The activities are connected to all parts in the back-end. Firstly, when the mobile app is started, a *Database* is created locally on the user's phone. Section 4.4 explains the database management and contents in more detail. The second part in the back-end is the gathered *Data* in the session. This can be data that is retrieved from the database such as information about shops or tags or temporary data that is not saved in the database such as a route along shops. The third element of the back-end is a *Global Class*, which is data that is globally known. This means that every activity in the application can set and get values that are saved in the *Global Class*. Further, the back-end contains the algorithms for the two *Recommender Systems*. Lastly, it consists of two *Route Actions*, needed for the actual route rather than the recommendation of a list of shops. One action is the calculation of a route, which is done using Google Maps Directions API. The other action is the parsing of the JSON object returned by the Google Maps API, returning a list of lists containing the latitude and longitude of the route. The set-up for the Android application can be seen in figure 4.1.

4.1.2 Web application

A web application of PESH is also implemented. This web application is hosted from a Virtual Machine (VM) on which Fedora 24 Server is installed. The VM serves webpages to a client based on the request of the client. The requests of the client are sent to the VM to Express, a web application framework for Node.js. The Express Server communicates with a Java (JDK 8) application via the node-java API, retrieving user-specific data. This data

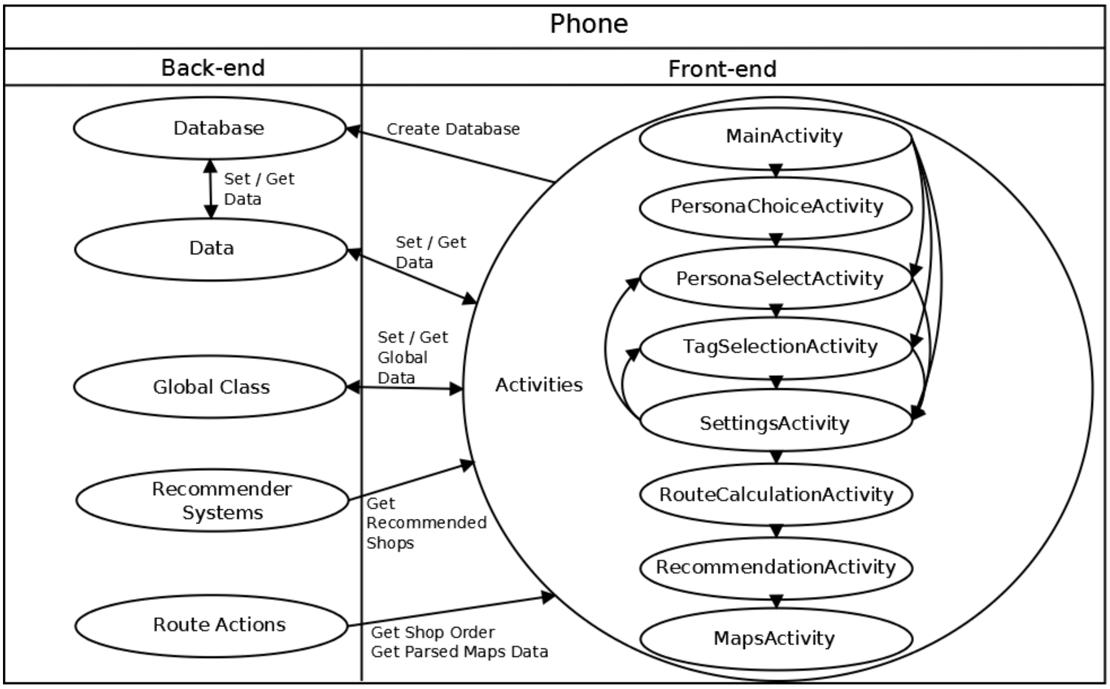


FIGURE 4.1: Set-up and lifecycle of Android application

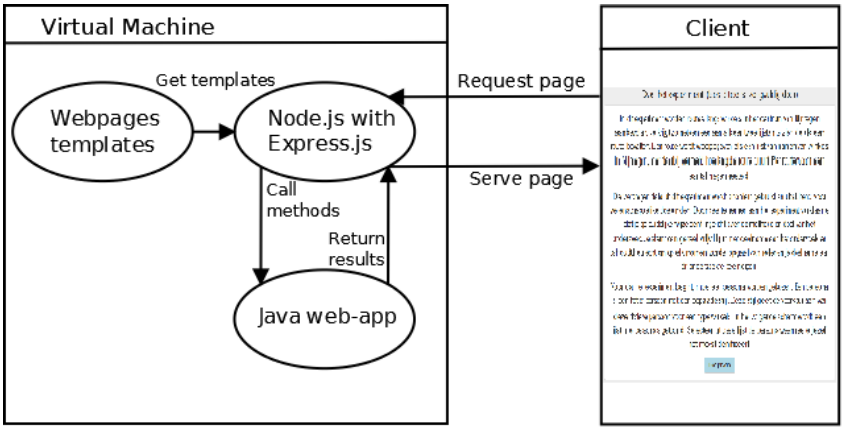


FIGURE 4.2: Set-up of web application

is combined with templates of webpages, thus creating dynamic webpages that contain user-specific data, such as a list of recommended shops based on a chosen persona. These dynamic webpages are then served to the client. The set-up for the web application can be seen in figure 4.2.

4.2 Walkthrough and Lifecycle

4.2.1 Android application

The purpose of the created mobile application is to show a route along shops optimized to the needs and preferences of the user. The user has to indicate two of his preferences before the route is shown. Firstly, a persona that the user identifies with the most and secondly, a time budget.

The assumption is made that the chosen persona is a fixed preference. This means that the persona that a user selects the first time, is usually the persona that the user would choose every time he uses the app. Therefore, once a user has chosen a persona, this choice is saved and loaded the next time the app is opened. Even though the persona is saved, it can be changed at every startup. The time budget is assumed to be variable and should thus be entered each time a route is searched for. Once both preferences are given, the recommended route given these preferences can be shown.

When opening PESH, a startscreen is shown followed by an introductory screen. This screen explains that a persona must be chosen that is most similar to the user. The user continues by clicking on a button. In the next screen, a list of all personas with their matching descriptions is shown from which the user needs to select one. After confirmation, the *settings* screen is shown. This is the first screen after the startscreen that will be shown to the user when restarting the app. It contains the selected persona and gives the user the opportunity to select a time budget in minutes, ranging from 30 to 240 minutes. When the value 0 minutes is selected, a pop-up comes into view with the message that a time budget should be selected. After confirmation, the *settings* screen is followed by a loading screen. When the recommended route is calculated, a list with all shops in the recommended route is shown. Shops can be selected, indicating it to be personal, and the selection can then be confirmed to get a new recommendation. The lifecycle of the Android application is shown in figure 4.1 and the walkthrough of the application can be found in Appendix C.1.

4.2.2 Web application

The web application is used for the experiment and thus built to evaluate the two recommender systems, *content-based* and *composite*. In this section, only the walkthrough and lifecycle will be explained, the experiment is discussed in chapter 5 and the evaluation in chapter 6.

The web application is put online on `http://pesh.cs.ru.nl` during the experiment. At the end of the experiment it has gone offline and thus cannot be accessed anymore. During the experiment, when going to this URL, a static webpage is shown with a textbox. This textbox contains an explanation of the experiment and serves as consent form. Users are told that the retrieved data will be used anonymously and solely for scientific purposes. Further, it is explained what a persona is. The user is then asked to choose a persona out of a list of personas that will be shown in the next screen.

The second screen thus contains a list of all personas. The user is asked to select the persona which (s)he can identify with the most. Per persona, the name, list of positive tags (likes) and list of negative tags (dislikes) are shown. When a user has selected one of the personas, the selection should be confirmed in order to go to the next webpage.

The webpage that comes up after the page with a list of all personas is a page that explains the first part of the experiment. When continuing to the next page, a cycle, consisting of two webpages, starts and is repeated once. The loop begins with two recommendations shown on the screen, one on the left side of the screen and one on the right side. For both recommendations the estimated time of the route is given. The user is asked to select the recommended route that is most personal. After confirmation, a new page appears containing a list of all shops that were recommended in the previous screen. The user is asked to select the shops that fit best to his or her personal interests. The cycle then starts over and is executed once more.

After the first cycle, an explanation is shown about the second part of the experiment. After this page, a new cycle starts, which again consists of two webpages. In total, the cycle is executed fifteen times. Here, on the first page, again two recommendations are shown. One is shown on the left side, the other on the right side and the estimated time of both routes is given. The user is again asked to select the recommended route that is most personal. The second page shows the list of all shops that were recommended in the two routes, and the user is asked to select the shops that fit his or her personal interests. This process is repeated another fourteen times. The walkthrough of the web application is pictured in Appendix C.2.

4.3 Ratings updates

In the previous section 4.2 it is discussed that feedback is gathered in the web application. This feedback consists of a persona, a preference for either a random-, content-based or composite recommender per iteration and a list of shops that fit the user's personal interests. The last type of feedback (on individual shops) is used to update the ratings for shops of the user. Each shop that is selected by the user is said to have *positive feedback*. All other shops have *negative feedback*. The ratings are updated according to the pseudocode described in Algorithm 1.

Algorithm 1: Ratings update

```
Function UpdateRatings(currentUser, oldTags, allTagNames, allShops,
 allShopNames, databaseHandler)
    List newTags ← createTagValues(oldTags, allTagNames,
      allShopNames, allShops);
    Ratings ratings ← new Ratings(allShops);
    ratings ← ratings.computeRatings(newTags);
    currentUser.setRatingList(ratings);
    databaseHandler.setUser(currentUser);
```

The list *oldTags* consists of pairs of tag names and a value. This value represents how much the tag fits the user's interests. The function *UpdateRatings* first updates these values and then computes the user's ratings for the shops. These ratings are set to the user and saved in the database.

4.3.1 Update tag values

The method *createTagValues()* takes four parameters: the old tag-value pairs, a list of all tag names, a list of all shop names and a list of all *Shop* objects. It returns a new list of tag-value pairs. The pseudocode for this method can be found in Algorithm 2.

Algorithm 2: Method for updating tag-value pairs

```
Function createTagValues(tags, allTagNames, allShopNames, allShops)
    double alpha ← 0.5;
    tags ← createNegTagValues(allShopNames, allShops,
      allTagNames, tags, alpha);
    tags ← createPosTagValues(allShopNames, allShops,
      allTagNames, tags, alpha);
    for $i \leftarrow 0$ to $tags.size()$ do
        tagValue ← tags.get(i);
        if tagValue.getRating() $> 1.0$ then
            tags.remove(i);
            tags.add(i, new TagValue(tagValue.getTag(), $1.0$));
            continue;
        if tagValue.getRating() $>= -1.0$ then
            continue;
        tags.remove(i);
        tags.add(i, new TagValue(tagValue.getTag(), $-1.0$));
    return tags
```

It can be seen that the old tag-value pairs are first adapted by the methods *createNegTagValues()* and *createPosTagValues()*. The pseudocode for these methods is given in Algorithm 3 and 4 respectively.

Algorithm 3: Method for updating tag-value pairs from negative feedback

```
Function createNegTagValues(allShopNames, allShops, allTagNames, tags,
  alpha)
    foreach shopName ← negFeedback do
        index ← allShopNames.indexOf(shopName);
        shopTags ← allShops.get(index).getTags();
        shopTagLength ← shopTags.size();
        foreach tagName ← shopTags do
            tagIndex ← allTagNames.indexOf(tagName);
            rating = tags.get(tagIndex).getRating();
            old ← tags.remove(tagIndex);
            tags.add(tagIndex, new TagValue(old.getTag(), rating -
              (alpha / shopTagLength)));
    return tags
```

These methods adapt the tag-value pairs based on the user's feedback. Firstly based on the negative feedback and afterwards based on the positive feedback. The methods iterate over all shops that have a negative feedback from the user. The index and assigned tags of the shop are retrieved as well as the amount of assigned tags. Per shop, a new iteration is started over these tags. The index and old value of the tag are retrieved. The old tag-value pair is removed and a new pair is added in its place. For each shop with negative feedback, the new value of the tag becomes $oldValue - (alpha/shopTagLength)$. For each shop with positive feedback, the new value becomes $oldValue + (alpha/shopTagLength)$. $oldValue$ is the value of the deleted tag-value pair, $alpha$ is a constant set to 0.5 and $shopTagLength$ is the amount of tags assigned to the shop. This means that the new value of the tag-value pair depends on the amount of tags a shop is assigned to. This is done because if a shop has many tags assigned to it, the chance that every tag is perceived negatively to the user is smaller compared to a shop with only one assigned tag. Therefore, the tag-value pairs are updated with respect to the amount of assigned tags.

Algorithm 4: Method for updating tag-value pairs from positive feedback

```
Function createPosTagValues(allShopNames, allShops, allTagNames, tags, alpha)
    foreach shopName ← posFeedback do
        index ← allShopNames.indexOf(shopName);
        shopTags ← allShops.get(index).getTags();
        shopTagLength ← shopTags.size();
        foreach tagName ← shopTags do
            tagIndex ← allTagNames.indexOf(tagName);
            rating = tags.get(tagIndex).getRating();
            old ← tags.remove(tagIndex);
            tags.add(tagIndex, new TagValue(old.getTag(), rating + alpha / shopTagLength));
    return tags
```

4.3.2 Compute ratings

The method *computeRatings()* takes one parameter: the newly created list of tag-value pairs. It returns a *Ratings* object, containing a list of all *Shop* objects and an array of all ratings of the participant for these shops. Algorithm 5 provides the pseudocode for this method.

Algorithm 5: Method to compute ratings

```
Function computeRatings(tags)
    ratings ← new double[allShops.size()];
    index ← 0;
    foreach shop ← allShops do
        shopTags ← shop.getTags();
        tagRatings ← 0.0;
        foreach tag : tags do
            if !shopTags.contains(tag.getTag()) then continue;
            tagRatings + = tag.getRating();
        nTotal ← shop.getTags().size();
        ratings[index] ← tagRatings / nTotal;
        ++index;
    allRatings ← ratings;
    return this;
```

The method *computeRatings()* is a method of the *Ratings* object. It adapts the array of all ratings of the participant and then returns itself, so that the new ratings can be set to the user and saved in the database. In *computeRatings()*, the ratings are updated per shop by iterating over all shops. The assigned tags are retrieved and compared to the tag-value pairs of the user. The values (for which apply that $-1 <= value <= 1$) for each assigned tag are summed. The final rating for a shop is then calculated by dividing this summation by the amount of assigned tags to the shop. This ensures that the ratings are normalized. The ratings are then set to the *Ratings* object which in turn is returned by the method.

4.4 Database

Data about the shops, tags and users is stored in an SQLite Database. In this section the management of the database and its content is discussed.

4.4.1 Database management

Android application

A local database is created with the helper class SQLiteOpenHelper from the *android.database.sqlite* package. This class helps creating and updating the database only when necessary. The database is created only during the first app startup. In later app startups, data can be retrieved from or writed to the database by calling the predefined public methods *getReadableDatabase()* and *getWritableDatabase()* respectively. The disadvantage of the used database is that it is saved locally. Therefore, data of user u_1 saved on mobile phone p_1 can not be used in the recommendation of shops for user u_2 using another phone p_2. Thus, collaboritave filtering, the process where preferences of similar users are used to make recommendations for other users, can not be applied. However, this study is limited to comparing the composite recommender system to a content-based recommender system.

Web application

Per user, three local databases are created with the *SQLite JDBC* library. *SQLite* is an SQL database engine and this library allows the access and creation of *SQLite* database files through the *JDBC* API. The *Java DataBase Connectivity (JDBC)* is a Java API through which a Java software can communicate with a database using SQL.

When a user visits `http://pesh.cs.ru.nl`, a unique session ID is created as well as three local databases. For both types of recommender systems (content-based and composite) a separate database is created to save feedback from users as separate instances. Moreover, an extra database is created for a third type of recommendations, namely random recommendations. This last type is needed to evaluate the experiments in order to answer our research questions. More about the exact evaluation can be read in chapter 6. The three databases are used to store the data that the user entered and to retrieve data to give recommendations.

4.4.2 Database contents

The database of PESH contains four tables. In table 4.1 the contents of these tables are listed.

TABLE 4.1: List of tables and their contents in the database of PESH

Table	**Contents**
User	Session ID: *Unique ID created when the app is started by the user* Persona ID: *Unique ID of assigned persona to the user* Positive Tags: *Tags that are assumed to be perceived positively by the user* Negative Tags: *Tags that are assumed to be perceived negatively by the user* Visited List: *List of shops visited by the user* Ratings List: *List of ratings for all shops by the user* Time Budget: *Time budget of the user for walking the route*
Personas	Persona ID: *Unique ID per persona* Persona Name: *Name of the persona* Positive Tags: *Tags that are perceived positively by the persona* Negative Tags: *Tags that are perceived negatively by the persona* Visited List: *List of shops (fictionally) visited by the persona* Ratings List: *(Fictional) Ratings of the persona per shop*
Shops	Shop Name: *Name of the shop* Shop Address: *Address of the shop* Shopping Time: *Estimated time spent in the shop* Shop Tags: *Tags assigned to the shop* Shop Similarities: *Similarities between current shop and all other shops* Latitude: *Latitude of the shop* Longitude: *Longitude of the shop* Shop Distances: *Distances between current shop and all other shops*
Tags	Tag Name: *Name assigned to the tag, e.g. Telecom or Accessories*

4.4.3 Creation CSV-files

The *personas*-table, *shops*-table and *tags*-table are filled with data read from CSV files. These files contain the required information about the personas, shops and tags. They were created using Python 2.7.11. We will discuss the creation of the CSV files per table.

Personas

The CSV file with personas consists of a *persona ID*, the *persona name*, a list of *positive tags* and a list of *negative tags*, a list of *visited shops* and a list of *ratings*. Fourteen personas are assigned *ID's* from 1 to 14. The *persona names* are added to the ID's.

A list of positive and negative tags is made for each persona. The list of *positive tags* and the list of *negative tags* are both a binary string with the length of the total amount of tags. For each positive tag, a 1 is put in the list of *positive tags* at the index of that tag in the list of *tag names* (see section *Tags* below). The same method is applied to each negative tag in the list of *negative tags*. For all other tags in both lists, a 0 is put on their corresponding indices. All $0's$ and $1's$ are followed by a semicolon, separating all indices.

By using the positive and negative tags of a persona, the list of *visited shops* is composed. This list consists of the shops that are fictitiously visited by the persona. It is set up similar to the list of positive and negative tags, with either a 0, indicating the shop has not been visited, or a 1, indicating that the shop has been visited. All values are separated by a semicolon and the indices of the list of *shop names* as described in the next section *Shops* are used. Each shop that has one or more assigned tags of either the positive or negative tags of the persona is set as 'visited' by the persona. All other shops are not visited.

The *ratings* list is also composed using the list of positive and negative tags and is described in section 3.2.2.

Shops

The CSV file with shops consists of the *shop name*, *shop address*, *estimated shopping time*, *shop tags*, *shop similarities*, the coordinates of the shop (*latitude* and *longitude*) and *shop distances*. From 341 shops, the *shop names* are added to the file, together with their *address* and *tags* as described in section 3.1.2. The *estimated shopping time* for each shop is 10 minutes as described in section 3.1.1.

The *shop similarities* $S(s_1, s_2)$ between two shops s_1 and s_2 are calculated as

$$S(s_1, s_2) = \frac{\sum_{tag \in tags} tag(s_1) \cdot tag(s_2)}{size(tags)}$$

where $tags$ is the binary list that indicates whether a tag is assigned to the shop and $size(tags)$ is the length of that list. $tag(s)$ is the binary value of the tag in $tags$ for shop s. Thus, the amount of tags that are assigned to both shop s_1 and shop s_2 is the measure for the similarity between any two shops s_1 and s_2.

The *shop distances* $D(s_1, s_2)$ between two shops s_1 and s_2 are retrieved using Google Maps Distance Matrix API. Not the actual distances between two shops is used, but the time to walk from shop s_1 to shop s_2 in seconds. These distances are calculated once for each pair of shops, therefore $D(s_1, s_2) = D(s_2, s_1)$.

Tags

The CSV file with tags consists of a list of all *tag names*. The full list of *tag names* can be found in Appendix A.

4.5 Persona-ization

Information about a user is obtained when he selects the persona that he can relate to most. With this information, *persona-ization* was implemented. User-specific ratings are calculated using the positive and negative tags of the persona. The ratings are a measure to predict a user his preferred shops. The rating $R(s_u)$ of a user u with positive tags pos_u and negative tags neg_u for shop s with an amount of $\#tags_s$ assigned tags is calculated as

$$R(s_u) = \frac{pos_u - neg_u}{\#tags_s}$$

4.6 Recommender systems

Two types of recommender systems are implemented, the *composite recommender system* and the *content-based recommender system*.

4.6.1 Composite Recommender System

The implementation of the *composite recommender system* is derived from the two approaches in Amer-Yahia et al., 2014. Firstly, *Produce-and-Choose (PAC)* where a set of bundles is produced and the best subset is chosen. The second approach is *Cluster-and-Pick (CAP)*, where compatible items are clustered, after which from each cluster a good bundle is picked. The field of use of these algorithms is originally search problems. However, in the current study, the algorithms will be used in a recommender system. Therefore, some minor changes have to be made. An important difference between the implementation of the algorithms in a search-related problem and our recommendation-problem is that in the latter the output should be only one bundle of shops. In the search-related problem in Amer-Yahia et al., 2014, a set of multiple, diverse bundles is given as output. We only recommend one bundle in our recommendation, namely a set of shops that together form a route. Because of the fact that there is only one bundle, the interbundle diversity is of no importance.

Parameters and functions

Table 4.2 lists parameters and functions used in both approaches. The descriptions of the parameters and functions are taken from Amer-Yahia et al., 2014. The *Value*-column describes the value that each parameter or function has in our application.

TABLE 4.2: Parameters and functions of the two approaches *PAC* and *CAP*, together with the description as mentioned in Amer-Yahia et al., 2014 and the value of the parameter as used in PESH.

Parameter	Description	Value
I	*Set of items*	List of shops in Nijmegen centre.
s(u,v)	*Pair-wise similarity function*	The *shop similarities* $s(u, v)$ between two shops u and v are calculated as $$s(u,v) = \frac{\sum_{tag \in tags} tag(u) \cdot tag(v)}{size(tags)}$$ where $tags$ is the binary list that indicates whether a tag is assigned to the shop and $size(tags)$ is the length of that list. $tag(s)$ is the binary value of the tag in $tags$ for shop s.
α	*Complementarity attribute*	Type of shop, determined by the associated list of tags per shop.
f	*Budget function*	$\sum_{S_i \in S} S_i(time) + walkingtime(S)$ S = a recommended route S_i = the i-th shop in recommended route S $S_i(time)$ = estimated time a user spends in shop S_i $walkingtime(S)$ = estimated time to walk recommended route S.
β	*Budget threshold*	By the user selected amount of time that he maximally wants to spend shopping. Ranging from 30 minutes to 240 minutes, with steps of 30 minutes.
k	*Amount of bundles*	Constant, set to 1.
γ	*User-defined scaling parameter*	Constant, set to 1.

Produce-and-Choose

The *Produce-and-Choose* algorithm is derived from the algorithm in Amer-Yahia et al., 2014. The method receives the parameters *I*, α, *f*, β, *k* and γ. The output is a set *S* of *k* valid bundles. Production method *BOBO-5* is favored over *C-HAC*, because the running time (for larger instances) is much higher for *C-HAC* (Amer-Yahia et al., 2014). The output of *BOBO-5* is a set of five valid candidate bundles, of which a graph is made. Our approach of the *choose*-phase is slightly different than the one described in Amer-Yahia et al., 2014. In the original approach iteratively the *least fitting* bundle is removed from the set of candidate bundles until *k* bundles are left. In our approach, all bundles except for the *highest scoring* bundle are removed. There is a subtle difference between these two approaches. The first, original, approach iteratively removes the bundle u that minimizes the function $\sum_{v \in S} w(u,v)$ with S being the set of candidate bundles and

$$w(u,v) = \frac{\gamma}{2(k-1)}(w(u) + w(v)) + (1-\gamma)\psi(u,v)$$

Here, $w(u,v)$ is the weight function between two bundles u and v, γ and k as defined in table 4.2. Further, for each candidate bundle B, $w(B) = \sum_{s_1,s_2 \in B} s(s_1,s_2)$ is a measure for the quality and cohesion of bundle B. $\psi(S_i,S_j) = 1 - max_{u \in S_i, v \in S_j} s(u,v)$ is the interbundle distance between bundles S_i and S_j. Naturally, the interbundle distance is of importance when there is more than one bundle. In our case, we only recommend one bundle of shops. Therefore, our approach does not have to take into account the second part of the formula for $w(u,v)$, $(1-\gamma)\psi(u,v)$. When removing the latter part, the formula becomes

$$w(u,v) = \frac{\gamma}{2(k-1)}(w(u) + w(v))$$

Because the user-defined scaling parameter γ is constant, it does not contribute to the outcome of $w(u,v)$ and is set to 1. Since $k = 1$, $w(u,v) = 1/(2(1-1)(w(u)+w(v)) = 1/0(w(u)+w(v))$ which is invalid because there is a division by zero. To be able to find the best bundle, we remove the invalid division $1/0$. Because the output is in our case not dependent of the other candidate bundles (with $k = 1$), we can also remove the addition of the measure for quality of the other bundles $w(v)$. Thus, we set $w(u,v) = w(u)$ for $k = 1$. This means that we iteratively remove the bundle in the candidate bundle set that has the lowest quality and cohesion. Intuitively, that is in line with what we want to recommend: the best set of shops, having the highest quality and cohesion. After iteratively removing the bundle in the candidate bundle set with the lowest quality and cohesion, one bundle remains that is recommended to the user.

To summarize, we have derived the algorithm

$$w(u,v) = w(u)$$

Minimization of the function $\sum_{v \in S} w(u,v)$ means in practice that the bundle with the highest quality and cohesion is recommended as a route to the user.

Cluster-and-Pick

The method *Cluster-and-Pick* has the parameters I, α, f, β and k as input. It gives a set S of k valid bundles as output. The first step is the *cluster*-step, in which k clusters are formed. From each cluster, a good bundle of shops is picked. In our case where $k = 1$, the clustering step would be unnecessary because only one cluster would be formed containing all the shops. This means that a good bundle is picked from all shops instead of from the clusters of shops. In this step, a bundle is created around each shop, where each bundle should satisfy the complementarity constraint and the budget constraint. The approach for the creation of the bundles around a pivot is the same approach that is used in the production-step in the *Produce-and-Choose* algorithm. Therefore, this method is not used in PESH and only the *Produce-and-Choose* method is implemented.

4.6.2 Content-Based Recommender System

A content-based recommender system tries to recommend items based on item-descriptions and preferences of a user (Pazzani and Billsus, 2007). Initially, a user gets assigned the positive and negative tags from the chosen persona. These tags are used to calculate the ratings of a user for all shops as described in section 4.5. The shops with the highest ratings are the ones that are recommended to the user. As a parameter, the content-based recommender system gets an integer n, indicating the amount of shops that should be recommended.

Chapter 5

The Experiment

In this chapter, the experiment that aims to answer our research questions as stated in section 1.2 is elaborated. For convenience, we will repeat the research questions here:

1. (a) Can *composite recommendation* be applied to a shopping environment?
   (b) Does *composite recommendation* outperform a *content-based recommender system* in a shopping environment, based on the above mentioned criteria?
2. (a) Can *persona-ization* be applied as a method to address the *cold start* problem in both *composite recommendation* and a *content-based recommender system*?
   (b) Are recommendations by both *composite recommendation* and a *content-based recommender system* perceived as more personal when applying *persona-ization* to address the *cold start* problem?

A list of stores in Nijmegen Centrum is obtained and assigned to one or more tags as described in section 3.1. A list of personas is composed as described in 3.2. Further, the web application discussed in chapter 4 is implemented.

The experiment consists of two parts, that will be explained separately. The first part of the experiment is designed so that the research questions *2a* and *2b* can be answered. In the second part, *composite recommender systems* and *content-based recommender systems* will be compared in order to answer research questions *1a* and *1b*. As mentioned in section 4.2.2, the web application is used to execute the experiment. The first screen shows the explanation of the experiment and serves as consent form. In Appendix C.2 the walkthrough of the web application is pictured per screen. The text that is shown on every screen can be read here.

5.1 Part 1 - Cold Start Problem

The first part of the experiment starts with the instructions for the participants to read. It is pointed out that in this part of the experiment two screens are shown two times. The first screen, which we will call *Route Screen*, consists of two lists with shops in Nijmegen, these are the recommended routes. The lists are shown side-by-side, as described in **experiment** On this screen, the most personal route should be selected. The estimated time is mentioned for both routes, however they should not be taken into account in this part. On the second screen, from now on called *Shop Screen*, a list with shops is shown, of which the shops should be selected that fit the participant's personal interests. As mentioned before, these two screens are shown two times, with the following order: *Route Screen*, *Shop Screen*, *Route Screen*, *Shop Screen*.

5.1.1 Composition of the *Route Screen*

The *Route Screen* is made up of two routes, shown side-by-side. One of those routes is always a route consisting of randomly chosen shops. The other route is either a composite recommendation or a content-based recommendation. Which type of recommendation is used in the first *Route Screen* is determined randomly. The other type is used in the second *Route Screen*. The position of the random route (either left or right) is also determined randomly for both screens.

It is made sure that both recommended routes consist of an equal number of shops. The composite recommender system cannot be assigned a static amount of shops, because it is dependent on the constraint *time budget*. With a certain time budget, e.g. 70 minutes, the route can consist of a different amount of shops in different cases. The random recommender system and the content-based recommender system can be assigned a static amount of shops. It is namely possible to compose a route of n randomly chosen shops and a route of the n best rated shops. Therefore, to make sure both recommended routes in the *Route Screen* have an equal amount of shops, the length of the route made by the composite recommender system is used as a measure. The length of the other recommender systems is equaled to this length.

To clarify this, suppose that the time budget is set to 70 minutes. The route recommended by the composite recommender consists of five shops. In each shop, it is assumed that the participant spends 10 minutes. The duration of walking the route is 15 minutes, giving a total of 65 minutes to visit all shops in the route. This amount fits in the time budget of 70 minutes, therefore the route is valid. Another route, containing six shops, with the assumption that the participant again spends 10 minutes in each shop and walking the route takes 15 minutes, has a total duration of 75 minutes which is more than the time budget allows. In this case, the route with four shops is recommended. For both the random- and content-based recommender, this would mean that $n = 4$.

The participant is asked to select the route that is most personal. When the participant confirms the selection, the feedback is saved. This feedback consists of the elements listed in table 5.1.

5.1.2 Composition of the *Shop Screen*

The *Shop Screen* contains a list of shops. In the previous screen, the *Route Screen*, two routes are recommended. The list in the *Shop Screen* consists of all shops that occur in one or both routes. Each shop can only occur once in the list. Participants are asked to select the shops that fit their personal interests.

When the selection is confirmed, feedback is saved for the recommender systems that recommended the routes in the previous *Route Screen*. For each

TABLE 5.1: Saved data from feedback in the *Route Screen*

Session ID
Name of the chosen persona
Type of recommender positioned left
Length of the left positioned recommender
Type of recommender positioned right
Length of the right positioned recommender
Recommender selected by the participant
Date and time

TABLE 5.2: Saved data from feedback in the *Shop Screen*

Session ID
Name of the chosen persona
Recommender name
Route length
Recommended shops names
Indices of the recommended shops
Names of shops that got a positive feedback (separated by a semicolon)
Names of shops that got a negative feedback (separated by a semicolon)
Date and time

recommender system the data that is saved given the participant's feedback is shown in table 5.2.

5.2 Part 2 - Composite versus content-based recommender systems

Similar to the first part, this part starts with instructions. This time, two screens are alternately shown fifteen times. In the first screen, which we will call the *Route Screen*, again two routes with shops in Nijmegen are shown side-by-side (**experiment**), one left and one right. Added to both routes is the estimated time the route takes. It is explained that the personally preferred route should be selected, taking into account a time budget of 70 minutes to walk a route. It is then stated that in the second screen, from now on called the *Shop Screen*, a list of shops is shown, of which the shops should be selected that fit to the participant's personal interests. As mentioned, these two screens are alternately shown fifteen times, making a total of fifteen iterations in this part.

5.2.1 Composition of the *Route Screen*

Whereas in the first part the *Route Screen* contained a route made by a random recommender system, in this part the *Route Screen* always contains a route made by the content-based recommender as well as a route made by the composite recommender. The position (either left or right) is determined randomly.

Further, the composition of the *Route Screen* is almost exactly the same as in the first part, described in section 5.1.1. Both routes have the same length and after confirmation of the selected route, the participant's feedback is saved. All elements listed in table 5.1 are saved as well as the iteration (1-15).

5.2.2 Composition of the *Shop Screen*

The *Shop Screen* is composed the same as in the first part of the experiment, described in 5.1.2. After confirmation of the selection of shops, the feedback is used to update the participant's ratings. The update of ratings is done according to the formula described in section 4.3. The new ratings are taken into account in the next iteration, so that the recommender systems adapt based on the participant's feedback. Further, as in the first part, the feedback is saved. All elements listed in table 5.2 is saved together with the iteration (1-15).

Chapter 6

Evaluation of the recommender systems and persona-ization

In the experiment, two parts are distinguished. Firstly, a part that aims to answer research questions 2a and 2b related to the *cold start* problem and *persona-ization*. Secondly, a part that aims to answer research questions 1a and 1b related to the appliance of composite recommender systems and its performance with respect to content-based recommender systems. The following sections discuss the evaluation of the experiment per research question.

6.1 Part 1 - Persona-ization

No actual evaluation is necessary to answer research question 2a. It is highly related with research question 2b. When the answer to research question 2b shows that recommendations by both the *composite recommender* and the *content-based recommender* are perceived as more personal than random recommendations, it follows that *persona-ization* can be applied as a method to address the *cold start* problem in *composite* and *content-based recommender systems*. If the recommendations are not perceived as more personal, the method *persona-ization* can be applied to the recommender systems, however it does not address the *cold start* problem.

Research question 2b can be answered by determining which recommended routes participants prefer. For this, we compare recommendations made by both recommender systems to random recommendations. In the first part of the experiment participants had to select their preferred route twice. In both cases, one route was recommended without *persona-ization*, so without taking the selected persona into account. The other route was either recommended by the *composite-* or *content-based* recommender using *persona-ization*. Both of these recommenders will be compared separately, to view the influence of *persona-ization* on both systems with respect to the *cold start* problem.

The set-up of the evaluation of research question 2b is similar to the one described in Thomas and Hawking, 2006. Where the field of research in Thomas and Hawking, 2006 is information retrieval (IR), in this research the method is used to evaluate different recommender systems. It consists of result sets in side-by-side panels. As described in chapter 5, our experiment has a two-panel interface with each panel showing one route.

The amount of preferences for the two systems with *persona-ization* (*composite* and *content-based*) and the amount of preferences for the random system without *persona-ization* are determined. For both systems with *persona-ization* it is measured separately whether they are preferred significantly more often than the random route. We use a binomial sign test ($p < 0.05$) to measure this significance because we test for differences between two pairs, either *content-based* versus *random* or *composite* vs *random*. Additionally, the binomial sign test is a distribution free test, making no assumptions about the probability distributions. Further, it is measured whether the combination of both systems is preferred significantly more often than the random route, again by using a binomial sign test ($p < 0.05$).

Besides testing whether recommender systems with *persona-ization* are preferred, we also compare the percentage of shops that is indicated to be personal. Using the wilcoxon test it is calculated whether there is a significant difference between these percentages for *persona-ization* and *no persona-ization*. The wilcoxon test is chosen because two paired groups of categorical data are compared. In addition, the wilcoxon test is also distribution free.

6.2 Part 2 - Composite versus content-based recommender systems

As stated in chapter 1, the performance of the recommender systems is based on two criteria. Firstly, the *quality*, e.g. how well the shops reflect the user's preferences. Secondly, the *time budget*, e.g. how well the recommended route fits within the user's time budget. We discuss the evaluation of these criteria in this section.

6.2.1 Quality

Similar to the relation between research questions 2a and 2b, there is a relation between research questions 1a and 1b. Only if *composite recommendation* can be applied to a shopping environment, we can answer whether it outperforms the *content-based recommender system*. Evaluating if *composite recommendation* can be applied to a shopping environment is simply done by checking the possibility to implement the algorithm described in Amer-Yahia et al., 2014.

To evaluate the performance of the *content-based-* and *composite* recommender, again the approach described in Thomas and Hawking, 2006 is used. The amount of preferences for both recommender systems are determined. Using a binomial sign test ($p < 0.05$) it is measured whether one of two systems significantly outperforms the other. This is both done per iteration and in total.

Also here, the percentage of shops that is indicated to be personal is compared for both recommender systems. Again, this is done per iteration and in total and the wilcoxon test is used to calculate whether the difference is significant.

6.2.2 Time budget

To test the performance of both recommender systems on the criterium *time budget*, the maximum time of a route recommended by each system will be given to show how well it fits to the set time budget of 70 minutes. The times of routes recommended by both systems are then compared using the wilcoxon test. This test will show whether there is a significant difference in route duration between the two systems.

Chapter 7

Results from the evaluations

The results of the experiment are separated in two parts. Firstly, the part where it is tested whether persona-ization contributes to a more personal recommendation. Secondly, the part where the composite recommender system is compared to the content-based recommender system. Both parts are again separated in two sub-parts that are equal in both the *persona-ization versus no persona-ization* part and the *composite recommender system vs content-based recommender system* part. In these sub-parts we will firstly look at the *preferred recommender system* and secondly at the *amount of shops selected as being personal*. Further, in section 7.2 we give the results about how well the recommended routes fit winthin the time budget of 70 minutes for both systems.

7.1 Persona-ization vs no persona-ization

A total of 39 participants completed the first part of the experiment. The participants are students familiar with the city Nijmegen. They were gathered using a Facebook post in groups of artificial intelligence and psychology students and by approaching students that are familiar with Nijmegen. The results are given in the following subsections.

7.1.1 Preferred recommender system

In this part, a recommender system using persona-ization (*persona recommender system*) is compared to a recommender system using no persona-ization, thus, as explained before, random (*random recommender system*). Both the composite recommender system and the content-based recommender system are assigned as instances of the *persona recommender system*. These systems are compared separately as well as combined with the *random recommender system*. Table 7.1 lists all comparisons between recommender systems with and without persona-ization and the amount of these recommender systems are indicated to be preferred. The table also shows whether there is a significant difference between both recommender systems and the appurtenant p-value.

7.1.2 Amount of personal shops

Table 7.2 lists for both *persona-ization* and *no persona-ization* the type of recommender system and the average percentage of shops that are indicated to be personal. It can be seen that all recommender systems *with* persona-ization perform better than the random recommender system *without* persona-ization. The average percentage of shops indicated to be personal with

Types recommender systems (versus random)	**Amount Preferred With Persona**	**Amount Preferred Without Persona (random)**	**Significant?**	**p-value**
Composite	30	9	Yes	0.001
Content-based	29	10	Yes	0.003
Combined	59	19	Yes	0.000006

TABLE 7.1: Preferred recommender systems, with persona-ization (composite and content-based) or without persona-ization (random)

With/Without Persona	**Type recommender systems**	**Average percentage of personal shops**
With persona	Composite	37.3
	Content-based	32.0
	Combined	34.6
Without persona	Random	16.0

TABLE 7.2: Average percentages of shops indicated to be personal *with persona-ization* and *without persona-ization*

persona-ization is significantly higher than without persona-ization ($p = 6.6 \cdot 10^-6$).

7.2 Composite recommender systems vs content-based recommender systems

The second part of the experiment is completed by 27 participants, their results are given in this chapter.

7.2.1 Preferred recommender system

The composite recommender system is compared to the content-based recommender system. In appendix C.3.1 a table with the number of times each system is preferred is given per iteration. For each iteration the p-value is added. The preferred systems per iteration is also shown in figure 7.1. Further, in table 7.3 the *total* number of times each system is preferred is shown. In total, the *content-based recommender system* performed significantly better than the *composite recommender system* ($p = 6.6 \cdot 10^-5$).

Recommender system	**Number of times preferred**
Composite	162
Content-based	243

TABLE 7.3: Preferred recommender systems, with persona-ization (composite and content-based) or without persona-ization (random)

Recommender system	**Average percentage of personal shops**	**p-value**
Composite	35.3	$1.3e - 19$
Content-based	56.8	

TABLE 7.4: Overall average percentages of shops indicated to be personal, composite versus content-based recommender systems

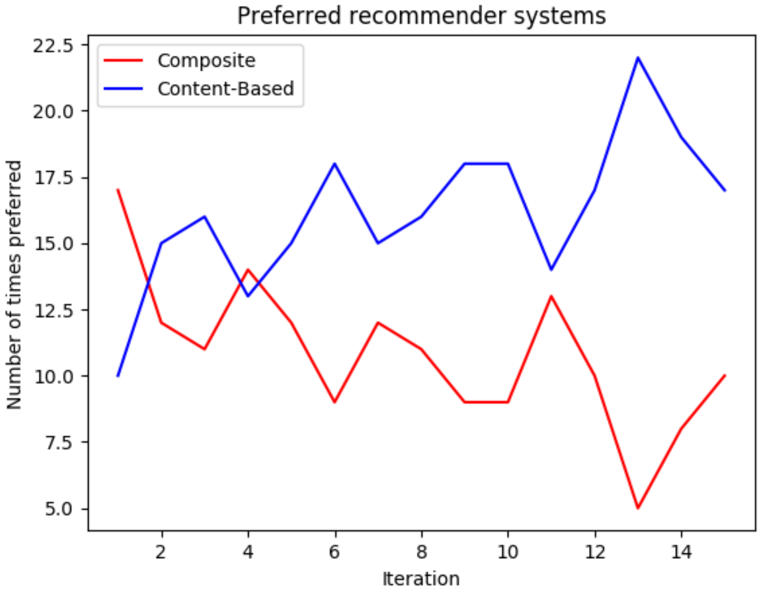


FIGURE 7.1: The number of times each recommender system is preferred per iteration

7.2.2 Amount of personal shops

Appendix C.3.2 shows a table with the average percentage of shops that were indicated as personal per recommender system for each of the fifteen iterations. Also, the overall averages per recommender system are given in table 7.4. Further, the data in the table in appendix C.3.2 is plotted in figure 7.2

Recommender system	**Maximum route duration**	**Average route duration**	**Standard deviation**
Composite	70	60.1	11.8
Content-Based	96	61.0	14.2

TABLE 7.5: Route duration per recommender system - Maximum, average and standard deviation

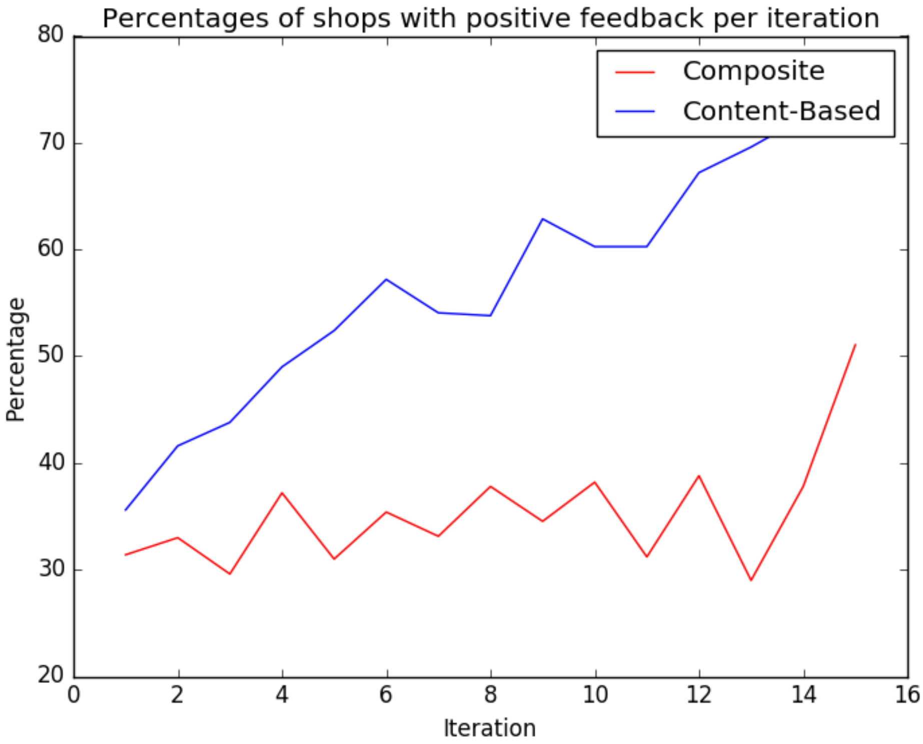


FIGURE 7.2: The average percentage of shops with positive feedback per iteration

7.2.3 Time budget

Table 7.5 shows per recommender system the maximum route duration, average route duration and standard deviation. The average route duration of the *content-based* recommender system is a bit higher than the average route duration of the *composite* recommender system, however the difference is not significant ($p = 0.08$).

Chapter 8

Conclusion and discussion

8.1 Conclusion

Chapter 7 shows the results from the experiment. Here, these results are used to answer the research questions.

8.1.1 Research questions

The performance of the recommender systems is based on the following two criteria:

1. *Quality*: how does the user perceive the quality of the route, e.g. how well do the shops reflect the user's preferences?
2. *Time budget*: how well did the recommended route fit within the user's time budget?

The research questions we want to answer are:

1. (a) Can *composite recommendation* be applied to a shopping environment?
   (b) Does *composite recommendation* outperform a *content-based recommender system* in a shopping environment, based on the above mentioned criteria?
2. (a) Can *persona-ization* be applied as a method to address the *cold start* problem in both *composite recommendation* and a *content-based recommender system*?
   (b) Are recommendations by both *composite recommendation* and a *content-based recommender system* perceived as more personal when applying *persona-ization* to address the *cold start* problem?

8.1.2 Persona-ization and the cold start problem

In the first part of the experiment, the influence of using *persona-ization* on recommendations was tested. The results show that routes recommended using persona-ization are preferred significantly more often than routes recommended without persona-ization ($p < 0.005$). This applies for both the composite and the content-based recommender system and thus also for the combination of both systems. Further, the average percentage of shops that is indicated as personal is significantly higher when using persona-ization ($p = 6.6 \cdot 10^-6$). From this, we conclude that persona-ization can be used as a solution to the cold start problem. With persona-ization, information about a new user is gained and used to increase the initial performance of the recommender system.

These results allow us to answer research questions 2a and 2b. It is possible to apply *persona-ization* as a method to address the *cold start* problem in both *composite recommendation* and a *content-based recommender system*. Also, the recommendations made with *persona-ization* are perceived as more personal when applying it to address the *cold start* problem.

8.1.3 Composite versus content-based

The second part of the experiment was set up to be able to answer research questions 1a and 1b. Firstly, *composite recommendation* can indeed be applied to a shopping environment (research question 1a). Our mobile application however could not handle calculations of routes with a duration of more than 90 minutes (using a *Nexus 5* simulator in Android Studio). Our web application calculates both the *composite* and *content-based* recommendations together in approximately 20 seconds. Our assumption is that this is mainly due to an inefficient implementation.

The results show that the *content-based recommender system* performs significantly better than the *composite recommender system* when we look at the total number of times each system is indicated as most personal ($p = 6.6 \cdot 10^{-}5$). At iteration level, of all 15 iterations only for the $13th$ iteration the *content-based* recommender system is preferred significantly more often than the *composite* recommender system ($p = 0.002$).

The average percentage of personal shops is also significantly higher for *content-based* than for *composite* recommendations. When looking at the individual iterations, it can be seen that for all iterations the average percentage of shops that is indicated to be personal is higher for *content-based* than for *composite* recommendations. The differences for some of the iterations are significant ($p < 0.05$). A trend is visible where in the beginning the average percentages are similar for both systems. For the *content-based* recommender system, the average percentage then shows an increase, whereas the average percentage of each iteration of the *composite* recommender system more or less stays the same, with a peak at the $15th$ iteration.

The results show that *composite recommendation* does not outperform a *content-based recommender system* in a shopping environment, but that it actually performs worse. In total, *content-based* recommendations are chosen significantly more often than *composite* recommendations and the average percentage of shops indicated as personal is significantly higher with *content-based* recommended routes. Also, a trend is visible where the average percentage of personal shops from *content-based* recommendations increase over time (meaning that the percentage is higher for a higher iteration) whereas this percentage remains more or less the same with *composite* recommendations. However, not for all iterations there is a significant difference between the two systems (iterations 1, 2, 3, 4, 8 and 15 show no significant differences). In the next subsection, we will elaborate upon possible explanations for these observations.

Further, the *time budget* was tested for both recommender systems. The results show that the duration of all routes recommended by the *composite* system is maximally 70 minutes, thus within the user's time budget.

This is obvious since we have given the *composite* recommender system this threshold as a constraint. Because we haven't given this constraint to the *content-based* recommender system, logically this system also recommends routes with a duration of more than 70 minutes. However, the difference in duration of the recommended routes for both systems is not significant ($p = 0.08$).

8.2 Discussion & Future work

The experiment in this study is *online*, meaning that participants did not actually walk the recommended routes but instead gave feedback about their preferences in an online experiment. Because the experiment is online, the validity in the real world is not tested. While significant differences between the two systems are found here, that might not be the case in the real world. There might be a discrepancy between *online* and *"offline"* (in the real world) because participants might not know one or more shops that are recommended. Thus, in order to form an opinion about whether the route is personal, they should visit these shops.

Thus, this work can be extended by testing its validity in the real world. Instead of an online experiment, participants could walk the two recommended routes and then indicate which route is the most personal. This could be done in two ways: either the whole experiment can be performed *offline* or only a part of the experiment. In the last case, there should not be a significant difference between the *online* and *offline* results for the *online* experiment to be a valid method.

Assuming the validity of our experiment, the findings in this project might mean that diversity in shops is not very important when shopping. The *composite* recommender system should give a more diverse set of shops than the *content-based* recommender system, because of its constraint where shops with more than 75% of overlap in tags cannot be recommended within the same set. In *content-based* recommended routes, there is no such constraint. Given that the latter system is preferred significantly more often, it is possible that people don't necessarily like routes with diverse shops. The fact that in approximately 76% of all *content-based* recommendations the aforementioned constraint is not satisfied seems to support this theory. It can be inferred from this that people prefer routes with multiple similar shops rather than routes with a diverse set of shops.

In future work, the *composite* and *content-based* systems can be tested in different environments than the shopping environment. Also, the efficiency of the implementation in the shopping environment can be improved, in order to decrease the time needed to calculate recommendations.

Since using *persona-ization* has a positive effect on the performance of the recommender systems, this method can be considered in future work as a solution to the cold start problem. Its use in recommender systems rather than search problems can be researched in more detail, for example by finding the optimal set of sample persona's for the user to choose from.

Because of our implementation of the experiment where we showed the participants an equal amount of shops for both recommender systems and a time constraint for only the *composite* recommender system, results could differ when setting this constraint for both systems. This could be done for the *content-based* recommender in two ways: either pick the highest k shops along which the route duration fits within the user's time budget or pick a subset of length k of the highest n shops, where $n >= k$, where the sum of the subset of length k has the highest rating and the route duration along thus subset of shops fits within the user's time budget. However, because there was no significant difference between the durations of the routes recommended by both systems, we assume this difference will be negligible.

Bibliography

Adomavicius, Gediminas and Alexander Tuzhilin (2005). “Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.6, pp. 734–749.

Amer-Yahia, Sihem et al. (2014). “Composite retrieval of diverse and complementary bundles”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.11, pp. 2662–2675.

Anagnostopoulos, Aris et al. (2016). “Tour recommendation for groups”. In: *Data Mining and Knowledge Discovery*, pp. 1–32.

Bennett, Paul and E Kiciman (2015). “Persona-ization: Searching on behalf of others”. In: *Proceedings of the International Workshop on Social Personalization & Search, co-located at SIGIR*, pp. 26–32.

Bota, Horatiu, Ke Zhou, and Joemon J. Jose (2015). “Exploring Composite Retrieval from the Users’ Perspective”. In: *Advances in Information Retrieval: 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29 - April 2, 2015. Proceedings*. Ed. by Allan Hanbury et al. Cham: Springer International Publishing, pp. 13–24. ISBN: 978-3-319-16354-3. DOI: 10.1007/978-3-319-16354-3_2. URL: http://dx.doi.org/10.1007/978-3-319-16354-3_2.

Bota, Horatiu et al. (2014). “Composite Retrieval of Heterogeneous Web Search”. In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW ’14. Seoul, Korea: ACM, pp. 119–130. ISBN: 978-1-4503-2744-2. DOI: 10.1145/2566486.2567985. URL: http://doi.acm.org/10.1145/2566486.2567985.

Davidson, Ian and SS Ravi (2005). “Agglomerative hierarchical clustering with constraints: Theoretical and empirical results”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 59–70.

Karypis, George and Vipin Kumar (1995). “METIS–unstructured graph partitioning and sparse matrix ordering system, version 2.0”. In:

Pazzani, Michael J and Daniel Billsus (2007). “Content-based recommendation systems”. In: *The adaptive web*. Springer, pp. 325–341.

Poslad, Stefan et al. (2001). “Crumpet: Creation of user-friendly mobile services personalised for tourism”. In: *3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477)*. IET, pp. 28–32.

Pruitt, John and Jonathan Grudin (2003). “Personas: practice and theory”. In: *Proceedings of the 2003 conference on Designing for user experiences*. ACM, pp. 1–15.

Rashid, Al Mamunur et al. (2002). “Getting to know you: learning new user preferences in recommender systems”. In: *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, pp. 127–134.

Resnick, Paul and Hal R. Varian (1997). “Recommender Systems”. In: *Commun. ACM* 40.3, pp. 56–58. ISSN: 0001-0782. DOI: `10.1145/245108.245121`. URL: `http://doi.acm.org/10.1145/245108.245121`.

Setten, Mark van, Stanislav Pokraev, and Johan Koolwaaij (2004). “Context-Aware Recommendations in the Mobile Tourist Application COMPASS”. In: *Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004. Proceedings*. Ed. by Paul M. E. De Bra and Wolfgang Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 235–244. ISBN: 978-3-540-27780-4. DOI: `10.1007/978-3-540-27780-4_27`. URL: `http://dx.doi.org/10.1007/978-3-540-27780-4_27`.

Thomas, Paul and David Hawking (2006). “Evaluation by comparing result sets in context”. In: *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, pp. 94–101.

Tkalcic, Marko et al. (2011). “Addressing the new user problem with a personality based user similarity measure”. In: *Proceedings of the 1st International Workshop on Decision Making and Recommendation Acceptance Issues in Recommender Systems*. Citeseer, p. 106.

Xie, Min, Laks Lakshmanan, and Peter T. Wood (2011). “CompRec-Trip: A composite recommendation system for travel planning”. In: *Conference: Proceedings of the 27th International Conference on Data Engineering*. ICDE 2011. Hannover, Germany. DOI: `10.1109/ICDE.2011.5767954`.

Yang, Wan-Shiou, Hung-Chi Cheng, and Jia-Ben Dia (2008). “A location-aware recommender system for mobile shopping environments”. In: *Expert Systems with Applications* 34.1, pp. 437–445.

Appendices

Appendix A

Tag Names

Dutch	English translation
Accessoires	Accessories
Antiek & Kunst	Antique & Art
Boek & Kantoor	Book & Office
Computerwinkel	Computerstore
Concept store	Concept store
Dameskleding	Women's Clothing
Dier & Plant	Animal & Plants
Elektronica	Electronics
Herenmode	Men's Fashion
Hobby & Sport	Hobby & Sports
Interieur	Interior
Juwelier & Optiek	Jeweler & Optics
Kado & Speelgoed	Gift & Toys
Kindermode	Childrenswear
Levensmiddelen	Foodstuff
Overige	Other
Persoonlijke verzorging	Personal care
Schoenen	Shoes
Speciaalzaken	Specialist store
Telefoonwinkel	Phone shop
Warenhuis	Warehouse
Baby & Kind	Baby & Child
Bakkerij	Bakery
Delicatessen	Delicacies
Doe-het-Zelf	Do-it-Yourself
Fiets	Bicycle
Hobby	Hobby
Juwelier & Horloge	Jeweler & Watch
Kantoorartikelen	Office supplies

TABLE A.1: Tag names and their translations in English

Dutch	**English translation**
Meubelen	Furniture
Mode accessoires	Fashion accessories
Natuurwinkel	Nature shop
Parfumerie & Drogisterij	Perfumery & Drugstore
Recreatie	Recreation
Running	Running
Slagerij	Butchery
Sport	Sports
Tuin	Garden
Zoetwaren	Confectionery
Boek & Tijdschriften	Book & Magazines
Cadeau	Gift
Dier	Animal
Huishoudelijk	Domestic
Kaasspecialist	Cheese specialist
Koffie & Thee	Coffee & Tea
Lingerie & Ondergoed	Lingerie & Underwear
Muziek, Films & Games	Music, Movies & Games
Optiek	Optics
Slapen	Sleeping
Speelgoed	Toys
Supermarkt	Supermarket
Tweedehands	Second-hand
Vloer, Wand & Raam	Floor, Wall & Window
Woondecoratie	Home decoration

TABLE A.2: Continuation of tag names and their translations

Appendix B

Personas - Positive and Negative Tags

Persona	**Positive tags**	**Negative tags**
Beauty & Fashion Lover (man)	Accessories; Men's fashion; Jeweler & Optics; Personal care; Shoes; Jeweler & Watch; Mode accessories; Perfumery & Drugstore; Lingerie & Underwear	Do-it-yourself; Hobby; Garden; Floor, Wall & Window
Beauty & Fashion Lover (woman)	Accessories; Women's clothing; Jeweler & Optics; Personal care; Shoes; Jeweler & Watch; Fashion accessories; Perfumery & Drugstore; Lingerie & Underwear	Do-it-yourself; Hobby; Garden; Floor, Wall & Window
Interior Decorator	Antique & Art; Concept store; Interior; Warehouse; Furniture; Domestic; Floor, Wall & Window; Home decoration	Do-it-yourself; Hobby
Book Lover	Book & Office; Book & Magazines	Running; Sports
Construction Lover	Do-it-yourself; Hobby; Garden; Floor, Wall & Window	Jeweler & Optics; Personal care; Perfumery & Drugstore; Jeweler & Watch
Animal & Nature Lover	Animal & Plant; Nature shop; Garden; Animal	Butchery
Electronics Lover	Computerstore; Electronics; Phone shop; Music, Movies & Games	Running; Sports
Health Freak	Nature shop; Running; Sports	Confectionery; Delicacies

TABLE B.1: Personas and their assigned positive and negative tags

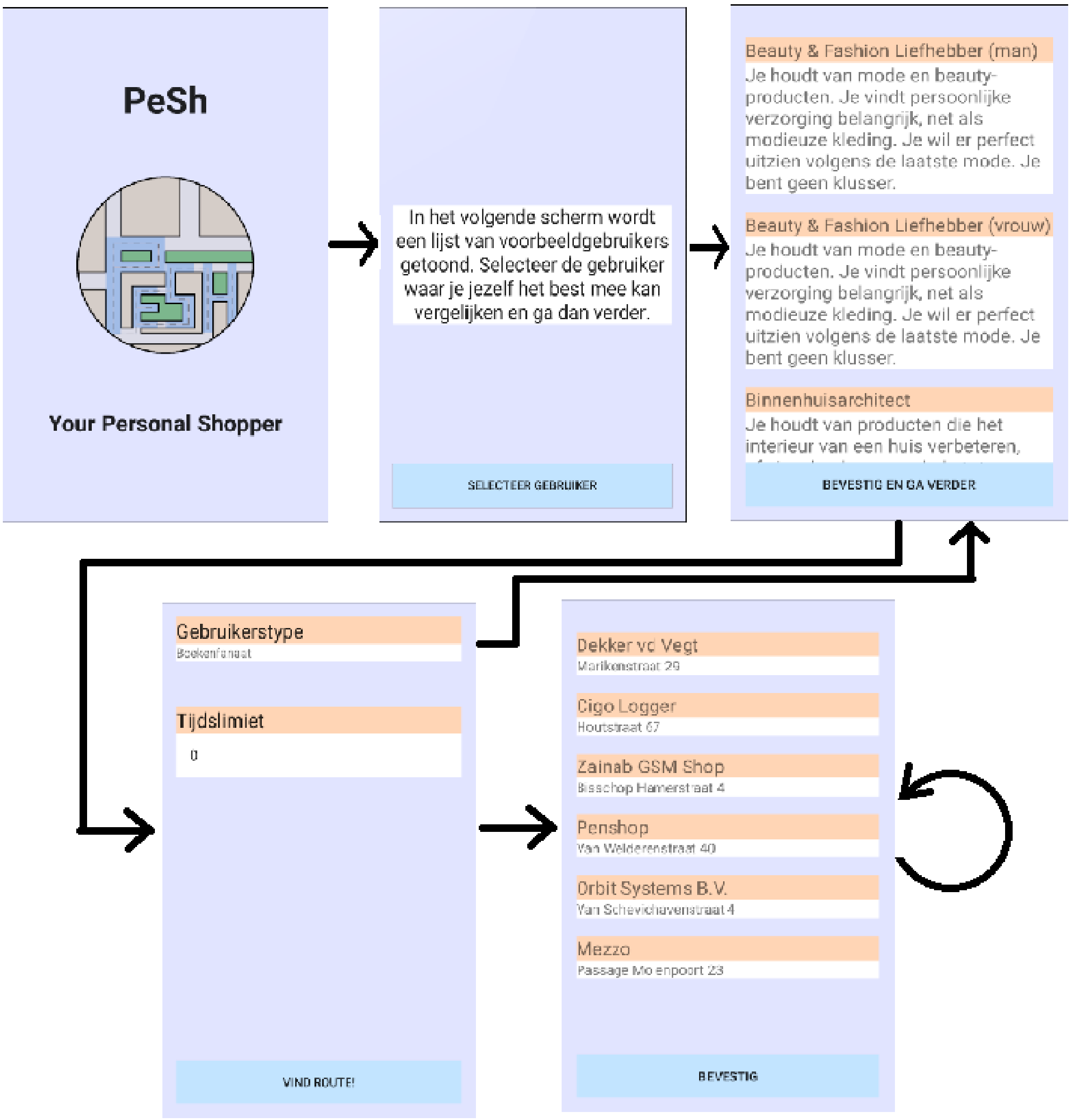
Persona	**Positive tags**	**Negative tags**
Kid (4-12)	Gift & Toys; Childrenswear; Baby & Child; Confectionery; Toys; Animal; Animal & Plant	Women's clothing; Men's fashion; Interior; Jeweler & Optics; Foodstuff; Personal care; Specialist store; Bakery; Do-it-yourself; Jeweler & Watch; Office supplies; Furniture; Nature shop; Perfumery & Drugstore; Antique & Art; Phone shop; Bicycle; Butchery; Garden; Domestic; Cheese specialist; Coffee & Tea; Lingerie & Underwear; Optics; Floor, Wall & Window; Home decoration
Art Lover	Antique & Art; Book & Office; Book & Magazines; Music, Movies & Games	Hobby; Do-it-yourself
Parents of young children	Women's clothing; Men's fashion; Childrenswear; Lingerie & Underwear; Baby & Child; Domestic; Warehouse	Furniture; Floor, Wall & Window
Jewelry Lover	Accessories; Jeweler & Optics; Jeweler & Watch; Fashion accessories	Optics; Computerstore; Electronics
Teenager (boy)	Computerstore; Electronics; Men's fashion; Childrenswear; Shoes; Music, Movies & Games; Confectionery	Women's clothing; Interior; Jeweler & Optics; Office supplies; Furniture; Butchery; Bakery; Domestic; Cheese specialist; Floor, Wall & Window; Home decoration
Teenager (girl)	Women's clothing; Childrenswear; Shoes; Confectionery; Accessories; Personal care; Perfumery & Drugstore; Fashion accessories	Men's fashion; Office supplies; Butchery; Bakery; Cheese specialist; Floor, Wall & Window; Electronics; Computerstore

TABLE B.2: Continuation of personas and their assigned positive and negative tags

Appendix C

Walkthrough PeSh

C.1 Android Application



In het volgende scherm wordt een lijst van voorbeeldgebruikers getoond. Selecteer de gebruiker waar je jezelf het best mee kan vergelijken en ga dan verder.

Beauty & Fashion Liefhebber (man)
Je houdt van mode en beauty-producten. Je vindt persoonlijke verzorging belangrijk, net als modieuze kleding. Je wil er perfect uitzien volgens de laatste mode. Je bent geen klusser.

Beauty & Fashion Liefhebber (vrouw)
Je houdt van mode en beauty-producten. Je vindt persoonlijke verzorging belangrijk, net als modieuze kleding. Je wil er perfect uitzien volgens de laatste mode. Je bent geen klusser.

Binnenhuisarchitect
Je houdt van producten die het interieur van een huis verbeteren,

Gebruikerstype
Boekenfanaat

Tijdslimiet
0

Dekker vd Vegt
Marikenstraat 29

Cigo Logger
Houtstraat 67

Zainab GSM Shop
Bisschop Hamerstraat 4

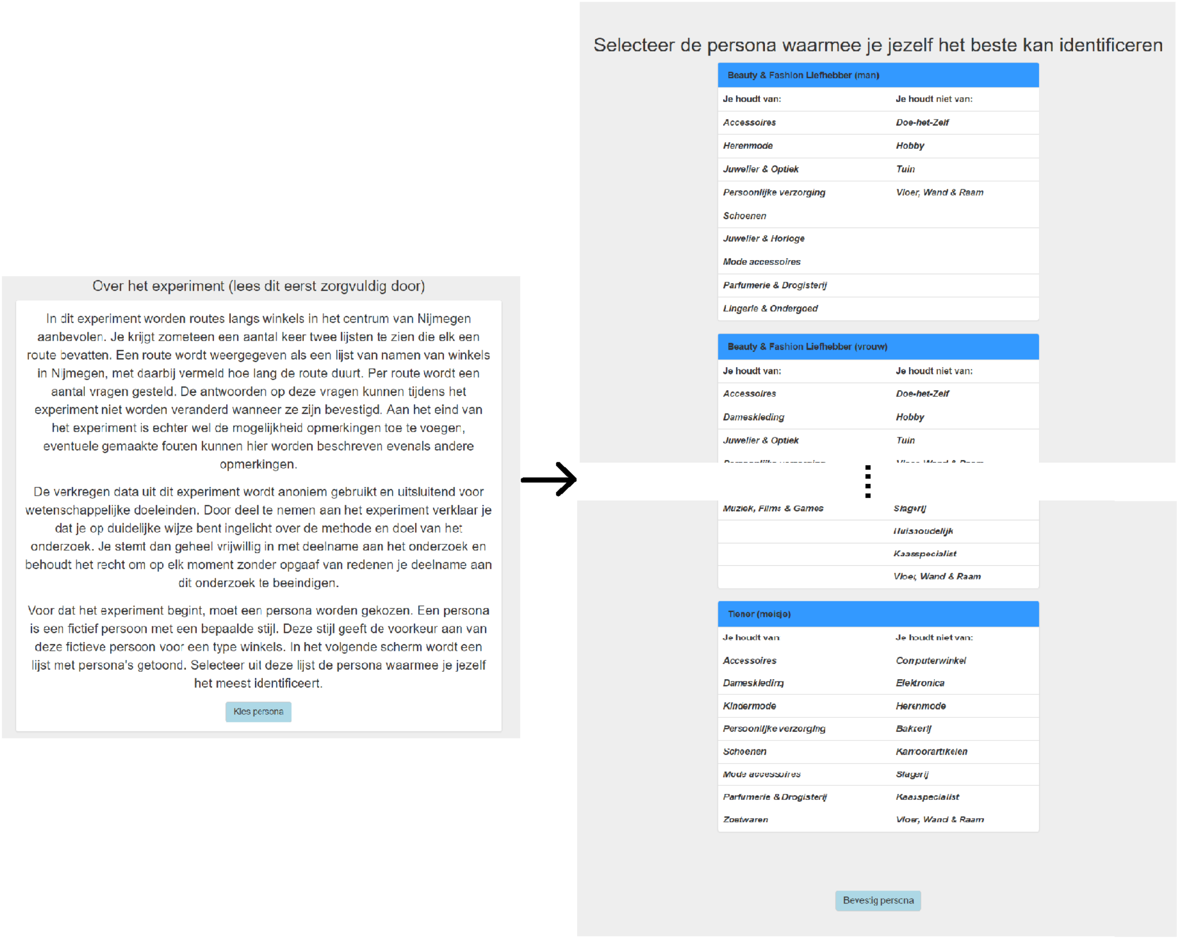
Penshop
Van Welderenstraat 40

Orbit Systems B.V.
Van Schevichavenstraat 4

Mezzo
Passage Molenpoort 23

FIGURE C.1: Walkthrough of the mobile application

C.2 Web Application



Over het experiment (lees dit eerst zorgvuldig door)

In dit experiment worden routes langs winkels in het centrum van Nijmegen aanbevolen. Je krijgt zometeen een aantal keer twee lijsten te zien die elk een route bevatten. Een route wordt weergegeven als een lijst van namen van winkels in Nijmegen, met daarbij vermeld hoe lang de route duurt. Per route wordt een aantal vragen gesteld. De antwoorden op deze vragen kunnen tijdens het experiment niet worden veranderd wanneer ze zijn bevestigd. Aan het eind van het experiment is echter wel de mogelijkheid opmerkingen toe te voegen, eventuele gemaakte fouten kunnen hier worden beschreven evenals andere opmerkingen.

De verkregen data uit dit experiment wordt anoniem gebruikt en uitsluitend voor wetenschappelijke doeleinden. Door deel te nemen aan het experiment verklaar je dat je op duidelijke wijze bent ingelicht over de methode en doel van het onderzoek. Je stemt dan geheel vrijwillig in met deelname aan het onderzoek en behoudt het recht om op elk moment zonder opgaaf van redenen je deelname aan dit onderzoek te beeindigen.

Voor dat het experiment begint, moet een persona worden gekozen. Een persona is een fictief persoon met een bepaalde stijl. Deze stijl geeft de voorkeur aan van deze fictieve persoon voor een type winkels. In het volgende scherm wordt een lijst met persona's getoond. Selecteer uit deze lijst de persona waarmee je jezelf het meest identificeert.

Selecteer de persona waarmee je jezelf het beste kan identificeren

Beauty & Fashion Liefhebber (man)

Je houdt van:	**Je houdt niet van:**
Accessoires	*Doe-het-Zelf*
Herenmode	*Hobby*
Juwelier & Optiek	*Tuin*
Persoonlijke verzorging	*Vloer, Wand & Raam*
Schoenen	
Juwelier & Horloge	
Mode accessoires	
Parfumerie & Drogisterij	
Lingerie & Ondergoed	

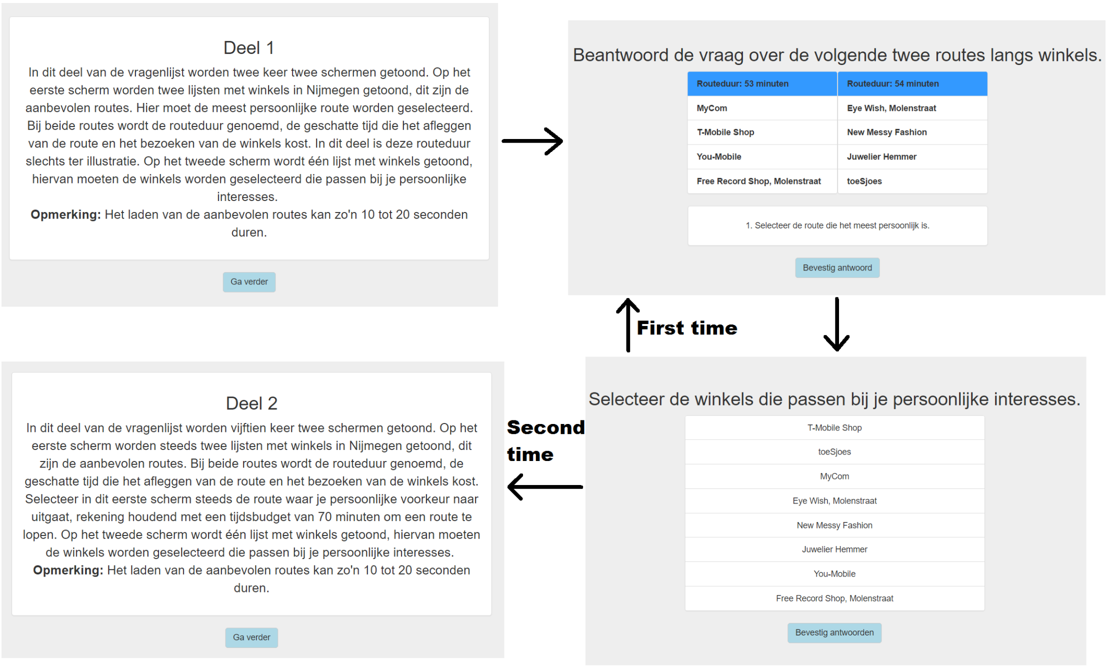
Beauty & Fashion Liefhebber (vrouw)

Je houdt van:	**Je houdt niet van:**
Accessoires	*Doe-het-Zelf*
Dameskleding	*Hobby*
Juwelier & Optiek	*Tuin*
Muziek, Films & Games	*Slagerij*
	Huishoudelijk
	Kaasspecialist
	Vloer, Wand & Raam

Tiener (meisje)

Je houdt van:	**Je houdt niet van:**
Accessoires	*Computerwinkel*
Dameskleding	*Elektronica*
Kindermode	*Herenmode*
Persoonlijke verzorging	*Bakkerij*
Schoenen	*Kantoorartikelen*
Mode accessoires	*Slagerij*
Parfumerie & Drogisterij	*Kaasspecialist*
Zoetwaren	*Vloer, Wand & Raam*

FIGURE C.2: First phase of the walkthrough of the computer application



Deel 1

In dit deel van de vragenlijst worden twee keer twee schermen getoond. Op het eerste scherm worden twee lijsten met winkels in Nijmegen getoond, dit zijn de aanbevolen routes. Hier moet de meest persoonlijke route worden geselecteerd. Bij beide routes wordt de routeduur genoemd, de geschatte tijd die het afleggen van de route en het bezoeken van de winkels kost. In dit deel is deze routeduur slechts ter illustratie. Op het tweede scherm wordt één lijst met winkels getoond, hiervan moeten de winkels worden geselecteerd die passen bij je persoonlijke interesses.
Opmerking: Het laden van de aanbevolen routes kan zo'n 10 tot 20 seconden duren.

Ga verder

Beantwoord de vraag over de volgende twee routes langs winkels.

Routeduur: 53 minuten	Routeduur: 54 minuten
MyCom	Eye Wish, Molenstraat
T-Mobile Shop	New Messy Fashion
You-Mobile	Juwelier Hemmer
Free Record Shop, Molenstraat	toeSjoes

1. Selecteer de route die het meest persoonlijk is.

Bevestig antwoord

Selecteer de winkels die passen bij je persoonlijke interesses.

T-Mobile Shop
toeSjoes
MyCom
Eye Wish, Molenstraat
New Messy Fashion
Juwelier Hemmer
You-Mobile
Free Record Shop, Molenstraat

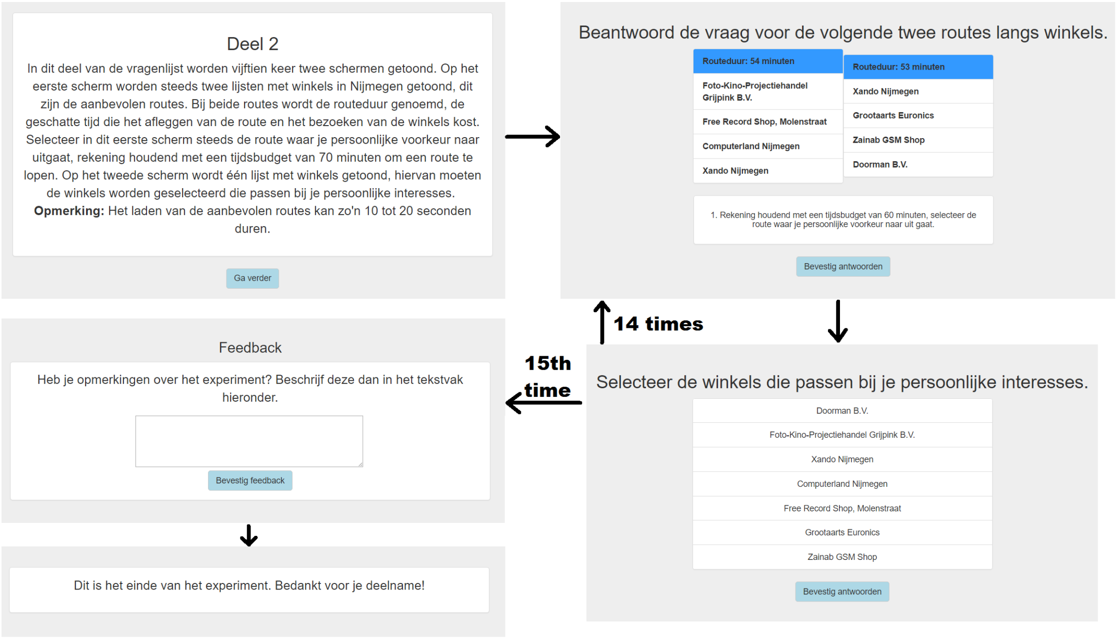
Bevestig antwoorden

Deel 2

In dit deel van de vragenlijst worden vijftien keer twee schermen getoond. Op het eerste scherm worden steeds twee lijsten met winkels in Nijmegen getoond, dit zijn de aanbevolen routes. Bij beide routes wordt de routeduur genoemd, de geschatte tijd die het afleggen van de route en het bezoeken van de winkels kost. Selecteer in dit eerste scherm steeds de route waar je persoonlijke voorkeur naar uitgaat, rekening houdend met een tijdsbudget van 70 minuten om een route te lopen. Op het tweede scherm wordt één lijst met winkels getoond, hiervan moeten de winkels worden geselecteerd die passen bij je persoonlijke interesses.
Opmerking: Het laden van de aanbevolen routes kan zo'n 10 tot 20 seconden duren.

Ga verder

FIGURE C.3: Second phase of the walkthrough of the computer application



Deel 2

In dit deel van de vragenlijst worden vijftien keer twee schermen getoond. Op het eerste scherm worden steeds twee lijsten met winkels in Nijmegen getoond, dit zijn de aanbevolen routes. Bij beide routes wordt de routeduur genoemd, de geschatte tijd die het afleggen van de route en het bezoeken van de winkels kost. Selecteer in dit eerste scherm steeds de route waar je persoonlijke voorkeur naar uitgaat, rekening houdend met een tijdsbudget van 70 minuten om een route te lopen. Op het tweede scherm wordt één lijst met winkels getoond, hiervan moeten de winkels worden geselecteerd die passen bij je persoonlijke interesses.
Opmerking: Het laden van de aanbevolen routes kan zo'n 10 tot 20 seconden duren.

Ga verder

Beantwoord de vraag voor de volgende twee routes langs winkels.

Routeduur: 54 minuten	Routeduur: 53 minuten
Foto-Kino-Projectiehandel Grijpink B.V.	Xando Nijmegen
Free Record Shop, Molenstraat	Grootaarts Euronics
Computerland Nijmegen	Zainab GSM Shop
Xando Nijmegen	Doorman B.V.

1. Rekening houdend met een tijdsbudget van 60 minuten, selecteer de route waar je persoonlijke voorkeur naar uit gaat.

Bevestig antwoorden

Selecteer de winkels die passen bij je persoonlijke interesses.

Doorman B.V.
Foto-Kino-Projectiehandel Grijpink B.V.
Xando Nijmegen
Computerland Nijmegen
Free Record Shop, Molenstraat
Grootaarts Euronics
Zainab GSM Shop

Bevestig antwoorden

Feedback

Heb je opmerkingen over het experiment? Beschrijf deze dan in het tekstvak hieronder.

Bevestig feedback

Dit is het einde van het experiment. Bedankt voor je deelname!

FIGURE C.4: Third phase of the walkthrough of the computer application

C.3 Composite versus Content-Based - per iteration

C.3.1 Preferred recommender system - per iteration

Iteration	**Recommender system**	**Number of times preferred**	**p-value**
1	Composite	17	0.248
	Content-based	10	
2	Composite	12	0.701
	Content-based	15	
3	Composite	11	0.442
	Content-based	16	
4	Composite	14	1.0
	Content-based	13	
5	Composite	12	0.701
	Content-based	15	
6	Composite	9	0.122
	Content-based	18	
7	Composite	12	0.701
	Content-based	15	
8	Composite	11	0.442
	Content-based	16	
9	Composite	9	0.122
	Content-based	18	
10	Composite	9	0.122
	Content-based	18	
11	Composite	13	1.0
	Content-based	14	
12	Composite	10	0.248
	Content-based	17	
13	Composite	5	0.002
	Content-based	22	
14	Composite	8	0.052
	Content-based	19	
15	Composite	10	0.248
	Content-based	17	

TABLE C.1: Preferred recommender systems per iteration

C.3.2 Amount of personal shops

Iteration	**Recommender system**	**Average percentage of personal shops**	**p-value**
1	Composite	31.4	0.556
	Content-based	35.6	
2	Composite	33.0	0.318
	Content-based	41.6	
3	Composite	29.6	0.054
	Content-based	43.8	
4	Composite	37.2	0.154
	Content-based	49.0	
5	Composite	31.0	0.010
	Content-based	52.4	
6	Composite	35.4	0.005
	Content-based	57.2	
7	Composite	33.1	0.019
	Content-based	54.1	
8	Composite	37.8	0.077
	Content-based	53.8	
9	Composite	34.5	0.000
	Content-based	62.9	
10	Composite	38.2	0.012
	Content-based	60.3	
11	Composite	31.2	0.011
	Content-based	60.3	
12	Composite	38.8	0.002
	Content-based	67.2	
13	Composite	29.0	0.001
	Content-based	69.6	
14	Composite	37.8	0.001
	Content-based	72.3	
15	Composite	51.1	0.075
	Content-based	72.7	

TABLE C.2: Average percentages of shops indicated to be personal per iteration, composite versus content-based recommender systems