KU LEUVEN

FACULTY OF SCIENCE

Visualization of music suggestions

A visual explanation system for collaborative filtering

Joris SCHELFAUT

Supervisor: Prof. E. Duval
Affiliation *(optional)*

Co-supervisor: *J. Klerkx*, *K. Verbert*
Affiliation *(optional)*

Mentor: *J. Klerkx*, *K. Verbert*
Affiliation *(optional)*

Thesis presented in
fulfillment of the requirements
for the degree of Master of Science
in Applied Computer Science

Academic year 2012-2013

Contents

Chapter 1

Introduction

1.1 Music suggestions and explanation systems

In their essence, recommender systems can be seen as filters applied on a large data sets. Ever since computer engineers started to develop this kind of systems, a wide range of algorithms have been designed and implemented to compute item recommendations[4, 35, 42, 43]; each of them with their own advantages and disadvantages.

To solve this problem, you will need some kind of explanation system that provides a reasoning to arrive at the results. An ambitious approach would be to explain each step of the recommendation algorithm, but this not always possible or desired. Indicating which of your tracks are closely related to the given recommendations, the system's confidence in the accuracy of the suggestions, et cetera, also help giving some additional context in explaining why a particular recommendation would be interesting[17]. Over the course of the last decade a wide range of explanation systems have been implemented. Many of these also use visualizations to explore user and/or item relationships[2, 5, 8, 15, 16, 41].

1.2 Thesis objective

The goal of this thesis is to design, implement and evaluate visualization and interaction techniques that will allow the user to gain insight into the recommendation process as well as actively steer the process. The elaboration of this thesis consists out of a literature study on the topic of visualization of music suggestions, and secondly a similar application that is designed and implemented[30].

1.3 The visual explanation system

The application created for this thesis is a page action Chrome Extension that injects *HTML* and *JavaScript* into the recommendations page of *Last.fm* at `http://last.fm/home/recs`. The application makes use of several *JavaScript* libraries, such as D3[1] and jQuery[2], as well as a specific JavaScript library by Felix Bruns[3] to facilitate the usage of

[1] A library using SVG, HTML and JavaScript[3]; available at: `http://d3js.org/`

[2] Available at: `http://jquery.com/`

[3] Available at: `https://github.com/fxb/javascript-last.fm-api`

the Last.fm API[4].

The application can be found in the Google Chrome web store[5].

1.4 Next chapters

The rest of this thesis text is organized as follows. Chapter 2 presents a literature study on recommender systems, visualization techniques, insight gaining and visual explanation systems. Chapter 3 tries to identify the users and how the application can be used. Chapter 4 describes the testing methodology and the different iterations. Chapter 5 looks at the different technologies that were used to develop the application and the architecture of the application, and discusses some of the specifics of the implementation. Chapter 6 concludes the thesis text. It provides an interpretation of the application's evaluation results, further conclusions, and a reflection on future work and opportunities.

[4]Available at: `http://www.last.fm/api`

[5]The SoundSuggest application can be found at: `https://chrome.google.com/webstore/detail/soundsuggest/jimmblcjmmjjfaklclmohcnabndlidmb`

Chapter 2

Literature study

This chapter describes the various components that are used in the development of this thesis. Section 2.1 looks at recommender systems and their rationale. Next we will look at visualization techniques and how these can be used to visualize the recommendation rationale in section 2.2. After this, insight gaining and human perceptual skills are investigated in section 2.3. We discuss how insight gaining can be evaluated and how the user is expected to use the visualization in a visual thinking algorithm. Finally we discuss and compare other visual explanation systems in section 2.4.

2.1 Recommender systems

A recommender system is a system that computes item suggestions for users based on a ratings of related items in the user's profile and/or history. Additional information can be incorporated into the recommendation algorithm to refine suggestions. Several categorizations of these techniques are proposed in literature [2, 4, 17, 35, 20].

One of the incentives behind creating recommender system is the 'long-tail phenomenon'[43]. This phenomenon can be explained as follows. Physical retail and warehouses can only keep a subset of all the available items in stock. These items are usually the most popular items on the market. Online vendors however, such as *Amazon*[1], can offer a vastly larger subset of these items to clients, including also less popular and/or less known items[43]. Typically the long-tail phenomenon is visualized in a graph in which items are ordered by their popularity on the horizontal axis against the popularity rating on the vertical axis, as can be seen on figure 2.1. Physical stores will offer only items in the first part of the graph, whereas the online vendors will also sell items from the remaining 'long tail' of the graph[43, 20]. Recommender systems then provide a means to find relevant items within this much larger range of items[43]. They enable to "connect supply and demand, introducing consumers to these new and newly available goods and driving demand down the tail "[1, 20].

Typical applications of recommender systems are product recommenders for online retailers, movie and music recommenders such as *Netflix*[2] and *Last.fm*[3], and news article recommenders in online news services[33, 43, 20].

[1] `http://www.amazon.com/`
[2] `http://www.netflix.com/`
[3] `http://www.last.fm/`

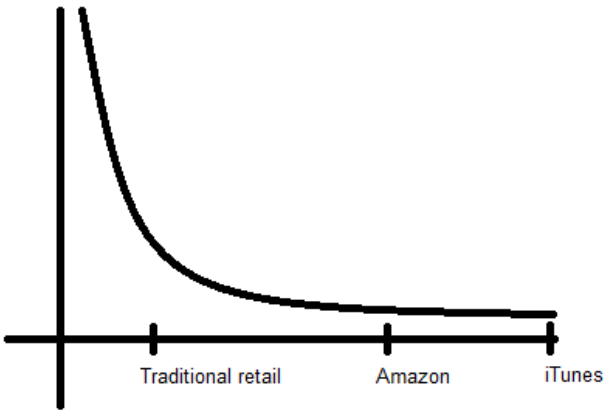


Figure 2.1: The long-tail: Items ordered by popularity are layed out against their popularity rating. Most of the items reside in the long tail of the graph. Companies such as Amazon can offer a vastly greater subset of the total item space.

2.1.1 Properties of recommender systems

In [18] and [46], Herlocker et al. and Shani et al. respectively, compare the performance of recommender systems. A number of metrics for recommendation algorithms are described among which the following properties are listed. We will also describe methods for testing these properties, as described in [46].

- **Accuracy**: The accuracy of item recommendations. There are three broad classes of prediction accuracy measures:
  - the prediction of the rating given by a user;
  - the prediction whether or not a user will actually use the item (for example adding to a queue) opposed to predicting the rating itself;
  - the prediction of a ranking among items rather than an explicit rating of each item independently.

  This kind of property is typically tested through an offline test on a training set, where parts of the recorded user profiles are hidden. The accuracy of the recommender system can then be determined by comparing the found recommendations to the remainder of the user profiles.

- **Novelty**: Novel recommendations are recommendations for items that the user did not know about. In a user study, users can be asked whether or not recommendations were new to them.

- **Serendipity**: Serendipity is a measure of how surprising the successful recommendations are. One can think of serendipity as "the amount of relevant information that is new to the user in a recommendation, or alternatively as deviation from the 'natural' prediction"[46]. Serendipitous recommendations usually carry a higher risk, as they fall further from the class of known preferences. To find out the

serendipity of recommendations, users can be asked directly if the recommendation was unexpected.

- **Diversity**: Diversity is generally defined as the opposite of similarity. To measure diversity content-based approaches can be used that compare recommended items.
- **Coverage**: One way to define coverage is the percentage of all items that are recommended to users during an experiment. Note that the cold start problem relates to coverage as it measures the coverage for a specific type of users, namely new users.

Many of these properties are closely related, as diverse recommendations that have a high coverage are likely to be serendipitous and novel. Also tradeoffs exist between properties; for example accuracy may drop as recommendations become more diverse[46]. Looking at the Long-tail phenomenon, coverage, serendipity, novelty and diversity tend to be important. In the context of music recommendation, users usually want to find new music[33].

2.1.2 A classification of recommendation algorithms

Based on classifications presented in [4] and [20], a categorization of different types of recommendation strategies can be identified. We will only discuss the two most prominent ones, namely collaborative filtering (CF) and content-based filtering (CB)[17, 43], and list some hybrid strategies. In the literature on recommender systems other general approaches that are commonly identified, are utility-based filtering, knowledge-based filtering, demographic filtering, and expert-based filtering[4, 2].

Collaborative recommendation

Collaborative recommendation aggregates item ratings by users. By establishing overlaps between ratings in the corresponding user profiles, the system generates new item recommendations[4, 17]. A typical user profile in a collaborative system consists of a vector of items and their ratings, that continuously augmented as the user interacts with the system over time[4].

For CF-based recommendation, there are two classes of entities: users U and items I. The data itself can then be represented by a utility matrix A. The entries $a_{i,j}$ of the utility matrix represent what is known about the degree of preference of user u_i and item i_j[43]. As can be seen in figure 2.2, the utility matrix will have many blanks as well. The goal of the recommendation algorithm is then to fill in the blanks[43].

In order to calculate the blanks, there is a variety of similarity functions that has been developed over the years. Often some sort of distance function is used to compute distance between profile vectors. If the distance is small, profiles will most likely have a high similarity[43]. The discussion of the mathematics behind each of the algorithms is beyond the scope of this thesis.

Last.fm is an example of a music recommender system that is based on the collaborative filtering approach[33].

		items			
		i1	i2	i3	i4
users	u1	a11	a12		
	u2	a21		a23	a24
	u3		a32		a34

Figure 2.2: The utility matrix A.

Content-based recommendation

Content-based recommendation learns a profile of the user's interests based on the features present in objects the user has rated. New recommendations can then be generated based on a similarity function on these features[4, 42].

When applying content-based filtering, the choice of similarity or classification function will have a significant impact on the quality of the recommendations. More importantly though, is the choice of features. To ensure good performance, these features should also be extracted easily from large quantities of data.

Depending on the type of item that is being recommended, different approaches can be applied to extract features and construct *feature vectors*. Textual information is often extracted using a technique called *stemming* that uses root forms capturing a common meaning behind groups of words. Tuples of root forms and TF.IDF (term frequency times inverse document frequency) scores are computed for each word[42, 43]. The words with the highest scores are the words that characterize the document[43]. A downside of stemming is that the process may cause the loss of contextual information for each word[42].

In [2] and [35] web crawlers are used to gather and extract features from online documents. Each property or feature is a 'bag of words' that can be used in a naive Bayesian text classifier. This way each item can be categorized and the profile can be 'learned'[35].

Tags are very useful as well. Although they can be generated from text, for complex objects such as images and music, tag generation relies on user input[43]. Nonetheless, emerging technologies such as the 'search by image' option introduced by *Google*[4], allow to retrieve web sites, documents and key words related to the given image[9].

Mathematical models for music also allow for feature extraction. Algorithms have been developed to classify music based on content features[34, 53]. There are various types of acoustic features that can be extracted. In [34] a distinction is made between rhythmic content features, pitch content features and timbral content features.

Hybrid recommendation

Hybrid filtering combines two or more recommendation algorithms[4]. In [4] a number of hybrid recommendation strategies are discussed. Robin Burke lists seven different approaches for combining recommendation algorithms. Each of these combinations also has its advantages and disadvantages. Not necessarily all combinations will be successful,

[4] `http://www.google.be/imghp?hl=nl&tab=Ti`

and not all of them have been implemented[4].

Based on the discussion in this paper by Robin Burke [4], hybrid systems for music recommendation can be built that use both collaborative filtering approaches and content-based approaches. For example Bostandjiev et al. [2] have built a music recommender that uses Wikipedia, a content-based source, Facebook, a collborative source, and Twitter, an expert-based source.

2.1.3 Challenges for recommender systems

Each recommendation technique has benefits as well as drawbacks. Some of these apply to all or most types of recommendation strategies, while others are only relevant to certain cases.

Cold start

Both CF and CB-based recommendation algorithms suffer from the ramp-up problem in one way or the other. The 'ramp-up' or 'cold start' problem (although they may refer to slightly different problems depending on the literature) is dual problem that encompasses two distinct, yet related problems as defined in [4]:

- **New User**: when a recommender system uses ratings by its users to compute item recommendations, it is hard to find neighbours for a user, who has a limited profile. As user profiles tend to build up over time, new users usually fall in this category.
- **New Item**: a new item will most likely not have that many ratings associated with it, and as a result will not be easily recommended. This 'new item problem' typically emerges when new items are constantly added to the system; for example when browsing a constant stream of news articles. When new articles are introduced, not many users have had the chance yet to rate these items. In the case of a news feed, an additional problem is that these items are short-lived, meaning that at some point these item profiles will most likely stop receiving any ratings at all.

Both of these issues translate themselves into a sparse regions in the utility matrix. It is worth noting that content-based recommendation algorithms suffer less from the *new item* problem, as these tend to rely on features that are inherent to the items themselves, rather than user generated content. This is one of the reasons hybrid approaches can provide a solution to collaborative filtering[4]. For example, in [35], content-based predictors are used to create pseudo-user ratings to reduce sparsity of the utility matrix, used in a collaborative algorithm.

Gray sheep

A problem that is typical of collaborative filtering is the 'gray sheep problem'[4, 17]. The gray sheep problem occurs when a user falls between different clusters of users that may have contradicting item ratings. As a result, it is hard to determine how to classify the user[4].

Black box

Another issue with recommendation systems is that these system often appear as 'black boxes' towards the end user. The complexity of the algorithms used, prevents the user from understanding the recommendation rationale[59]. This problem decreases the acceptance by the user of item suggestions. One of the solutions for this problem, proposed by Herlocker et al. in [17], is to provide an explanation system, i.e., the white box, on top of the recommender system that explains the recommendation process. This can be done through providing a transcript of the system's reasoning or through visualizations[17].

In this thesis we will focus mainly on this problem in the context of collaborative music recommendation. We will try design, implement and test a new visual explanation system in an effort to overcome this problem.

2.2 Information visualization

In this section different aspects of information visualization that were used to visualize the recommendation rationale are highlighted. First types of data are discussed, next we will look at visual encodings for these types of data. Finally we will discuss some visualization, interaction and data reduction techniques that were used in the *SoundSuggest* application.

2.2.1 Types of data

Information visualization has been focusing on on data sets that lack inherent spatial semantics, thus posing a challenge to map the abstract data onto a two-dimensional screen space[25]. Within this category of data, still different types of data can be distinguished and their characteristics will have an influence on the type of visualization.

Tables of data consist out of rows, representing items, and columns, representing the data dimensions, or 'attributes'. The number of dimensions is referred to as the dimensionality of the data set[25]. There are three different kinds of dimensions, namely[47]:

- **Quantitative**: numerical data on which arithmetic can be applied. For example playcount of a particular track, the duration of a track, the number of artists that two users have in common, et cetera.
- **Ordered**: an enumeration that has a definite order. For example ratings such as 'good', 'average', and 'bad'.
- **Categorical**: data that has no specific ordering, and is distinguished by name only. For example composer names, band members, artist tracks, users who like a particular item, et cetera.

In the utility matrix each column corresponds to an item and a row to a user. The entries of this matrix are typically quantitative data, while item and user names are categorical data.

Relational data on the other hand consists out of nodes and links or 'edges'[25, 47]. Both nodes and edges can have associated attributes. These attributes can again be either of quantitative, ordered, or categorical nature.

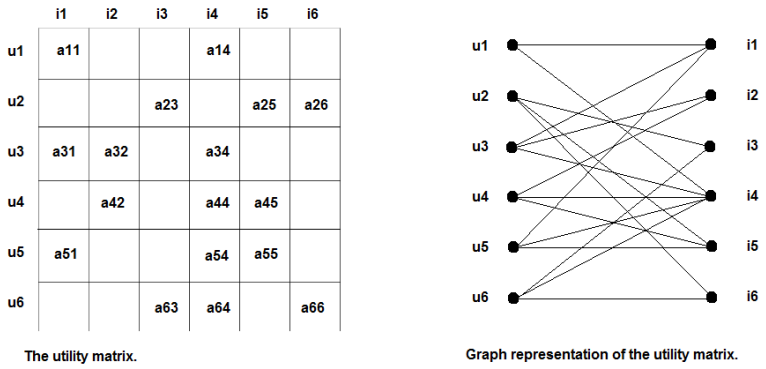


Figure 2.3: Transforming the utility matrix into a dual graph: two distinct sets of nodes, users and items, only share edges between nodes of different sets.

The underlying structure of collaborative filtering, the utility matrix, can be interpreted as a *dual graph*. This is a graph $G(V, E)$ for which $V = U \cup I$ such that $U \cap I = \emptyset \wedge E \subseteq U \times I$[6]. Each non-blank entry in the utility matrix will then correspond to an edge. Figure 2.3 shows how a matrix is transformed into a dual graph.

When applied to the context of collaborative filtering, the set of nodes U corresponds to the set of users, and the other set of nodes I is set of items. In conclusion this means that there only exist edges of that go from an item to a user or from a user to an item.

2.2.2 Visual encoding and visual channels

Visual encoding is defined as the mapping of data set attributes to a visual representation. The choice of visual encoding is one of the central problems in the visualization design[47].

Visual encoding takes place through *visual channels*. A visual encoding corresponds to a graphical element, or 'mark'. Examples of visual channels are spatial position, color, size, et cetera. The dimension of the mark may vary: a point is a zero-dimensional mark, a line a one-dimensional one, an area a two-dimensional one and so on.

A visual encoding has the following characteristics, as described in[47]:

- **Distinguishability**: the ability of a user to distinguish between visual encodings;
- **Seperability**: Separable visual channels are opposed to integral visual channels, which are focused together on a pre-conscious level. Separable visual channels are safe to use for encoding multiple dimensions;
- **Pop-out**: selecting a channel and make it visually stand out from all the others.

There is a variety of possible visual channels that a visualization designer can turn to in order to create a visual encoding, such as colour, spatial position, size, shape, orientation, and so on. The performance of the visual encoding (through a visual channel) depends on the type of data, i.e. quantitative, ordered or categorical [47]. Figure 2.4 gives an overview of the performance for each category, adapted from [47]. Note that spatial position is the most accurate for each data type[47].

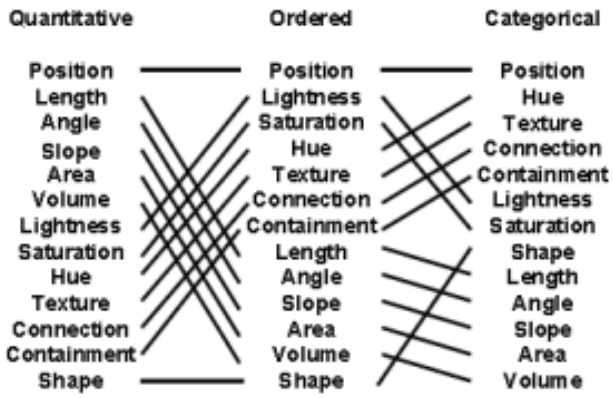


Figure 2.4: Visual encoding performance for each data type, ordered from best to worst.

Colour

In [47] colour is considered in terms of three separate channels: *hue*, *saturation* and *brightness*. This allows for different encodings. Just like for most visual channels, the choice of the channel (hue, saturation or brightness) depends heavily on the type of data.

For categorical data, hue can be successfully applied, keeping in mind its small range. An important remark is that roughly 10% of men is red-green color deficient. If a coding uses red and green, it may be wise to apply redundant coding using lightness or saturation in addition to hue [47].

In the *SoundSuggest* application, items in a list of neighbours are highlighted using the hue visual channel to distinguish between different groups of items.

Spatial layout

Spatial layouts form other visual channels. Although these tend to be the most accurate, spatial layouts in two and three dimensions have several weaknesses; two of these are[47]:

- **Occlusion**: Parts of the data set become hidden by others. In the case of the mapping of abstract dimensions onto spatial positions, understanding the details of a three-dimensional visualization may be challenging, even if the user is allowed to change viewpoints. For the *SoundSuggest* application, edges may overlap.
- **Text in arbitrary orientations**: Special care has to be taken with text, as it may become very hard to read depending on the orientation. In the *SoundSuggest* application artist names are turned according to their position on the circle layout.

To overcome limitations of visual channels, various visualization, interaction, and data reduction techniques may be applied[25, 47, 57]. Before discussing this any further, the following section will first take a closer look at graph drawing.

2.2.3 Graph-based visualization

Relational data is data that has an inherent relation among its elements[47]. The graph drawing problem describes the problem of how nodes and edges are visualized on a display[19].

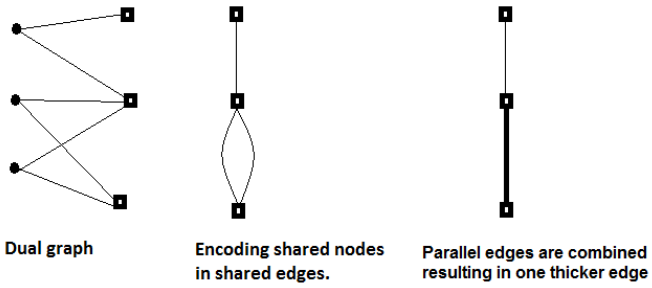


Figure 2.5: A row reduction operation on each pair of edges in a dual graph will result in a dimensionality reduction where one set of nodes is removed from the graph. An additional data reduction can be achieved by clustering edges into a thicker edge. Edge thickness then depends on the number of edges involved.

Scalability is one of the central issues with graph drawing, as graph size poses several important challenges[19]. The following issues are of course closely related to the general problems of visualization design discussed in section 2.2.2:

- **Viewability and discernability**: even if it is possible to layout and display all the elements, it may become impossible to discern between nodes and edges;
- **Performance and responsiveness**: graph layout algorithms may be relatively complex, and a large number of nodes and edges may become a bottleneck for performance, especially in interactive applications that require reasonable responsiveness;
- **Usability**: apart from problems with discernability, also information overload may occur. It is known that detailed analysis of data in graph structures is easiest when the displayed graph is small.

As graph size is one of the biggest issues of graph visualization, techniques have been developed to apply data reduction techniques on graphs[19, 47]. Apart from the traditional distortion and data transformation techniques, there are some techniques that are specific for graph-based visualization.

Data and dimensionality reduction

Based on a visualization design by Valdis Krebs in [50], a dimensionality reduction can be performed on the dual graph, by keeping only one set of nodes and representing the other set of nodes as implicit information in the edges. Figure 2.5 shows an example of this idea of 'row reduction'. In Krebs' visualization the items, books purchased from the Amazon web store in this case, were retained. In the resulting graph, two items would share an edge if a user bought both these items. The thickness of the edges represented the number of users that where linked to these items[29, 50].

The *SoundSuggest* application does not use the final step of Krebs' graph design. Instead parallel edges are retained to keep a direct link between user and edge. In the resulting visualization of the CF-based recommender, a quantification of the similarity between users can then be established by counting parallel edges between items that

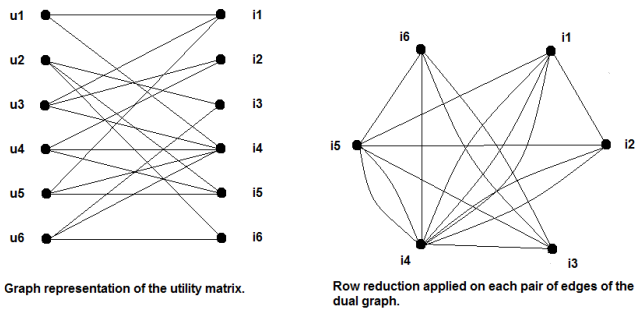


Figure 2.6: Row reduction applied on the graph in figure 2.3.

occur in neighbouring profiles. Figure 2.6 shows how the dual graph from figure 2.3 is transformed into a circular graph layout with the remaining item nodes, similar to the visualization in figure 2.7.

As it is unlikely that the whole user profile can be shown in the graph while avoiding visual clutter, the active user's favourite items are used to give a representation of the active user's profile. This way the user can still directly compare him/herself with neighbouring profiles.

Clutter reduction

Based on a survey by Herman et al. [19], and papers by Holten [21], and Holten et al. [22], the following gives an brief overview of two techniques: *clustering* and *edge-bundling*.

Clustering is the process of discovering groupings or classes in data based on chosen semantics, i.e., structure, or content. An example of structure-based approach is to bundle groups nodes that have certain number of edges between them. For content-based approaches, edges or nodes with similar attribute values can be clustered.

Due to the sparsity of the utility matrix, we might have to cluster multiple neighbouring profiles in a single node to ensure adequate connectivity within the resulting graph.

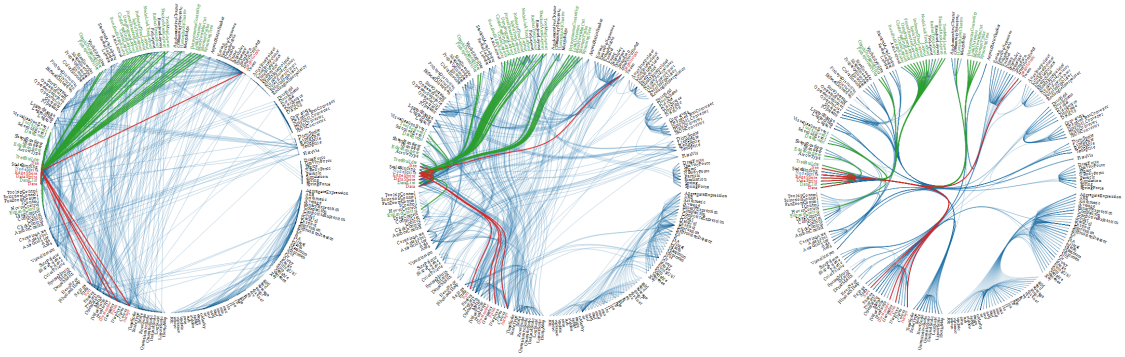
Edge-bundling is a technique to visualize compound graphs[21]. Edges are modelled as "flexible springs that can attract each other, similar to how electrical wires are bundled within a network"[22]. Figure 2.7 gives an overview of the different results for varying bundling strengths. It shows how edges are drawn closer towards each other, reducing visual clutter, and highlighting relationships between nodes.

Figure 2.8 shows how this applied on the graph from figure 2.6. From a graph drawing perspective we managed to reduce visual clutter by reducing the number of displayed items and applying edge-bundling[19, 21].

2.3 Gaining insight into interactive visualization

The goal of the *SoundSuggest* application is to allow the user to gain insight into the system through an interactive, visual explanation system. Two specific questions arise:

- how does a human gain insight?



(a) Low edge bundling strength.

(b) Medium edge bundling strength.

(c) High egde bundling strength.

Figure 2.7: Hierarchical edge bundling. Taken from `http://mbostock.github.io/d3/talk/20111116/bundle.html`. By increasing the bundling strength, edges will be drawn towards each other, clearly marking pathways between endpoints.

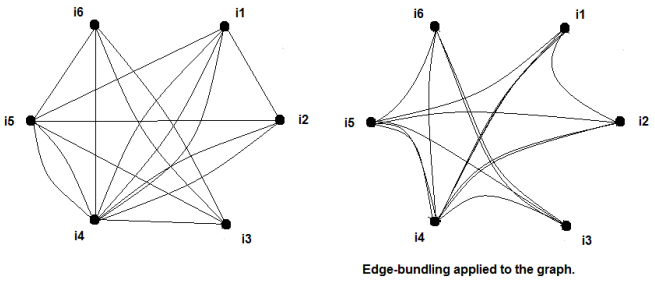


Figure 2.8: Edge bundling applied on the graph in figure 2.6.

- are there any human limitations imposed on the design of an interactive visualization?

The following subsections try to establish an answer to these questions. Most of the ideas in these subsections are drawn from a papers by Yi. et al. [58], Chris North [40], Klein et al. [27, 28], and a book by Colin Ware [57].

2.3.1 Insight gaining

In [40] it is argued that insight is not a well-defined term. A formal definition might be too restrictive to capture its essence, and yet too broad to be useful. To quantify insight, [40] and [58] list characteristics that allow a finer evaluation:

- **Complex**: insight is complex in the sense that it involves large amounts of data, cf. the input data of a recommender system, that form cognitive constructs, rather than individual units.
- **Deep**: insight is self-generating in a way, as insight provides a starting point for insight on the next level. Users may apply previously gained insight into a certain recommendation on other recommendations.
- **Qualitative**: insight is subjective, uncertain and can have multiple levels of resolution;
- **Unexpected**: insight is usually unpredictable, serendipitous and creative;
- **Relevant**: insight is deeply embedded in the data domain: it gives data meaning as it connects data to the existing domain knowledge: the user has to give meaning to seeminly random suggestions.

The quality of insight can then be determined by quantifying each of these characteristics[40]. North describes methods to evaluate insight gaining through visualizations, such as usability testing, heuristic evaluation, cognitive evaluation, and controlled experiments on benchmark tasks[40].

Chris North notes that controlled experiments suffer from four fundamental problems that may hinder effective evaluation of previously listed characteristics of insight. Such experiments are[40]:

- **Predefined**: Following specific, predefined instructions leave little room for unexpected insight.
- **Limited in time**: Short task times leave little room for deep insight.
- **Definitive**: Multiple choice questions leave little room for quantative insight.
- **Superficial**: Answers are concise, leaving little room for complex and relevant insight.

In this thesis we will focus on the think aloud technique instead. This is part of an alternative method described by Chris North that includes three key innovations[40]:

- an open-ended protocol;
- a qualitive insight analysis;
- an emphasis on domain relevance.

In this technique users are free to explore the data. By applying the think aloud protocol, the user's insight's can be captured. In order to increase domain relevance, the test users should be users from the target audience. Domain experts can then provide "critical metrics for the value of importance of the reported insights in the domain"[40].

Sensemaking

Sensemaking plays an important part in insight gaining [58]. The definitions for sensemaking may vary. We adapt the definition presented by [27] and [58].

In [27] sensemaking is looked at from a psychological perspective, a perspective of human-centered computing, and the perspective of naturalistic decision making. Sensemaking is then defined as follows: "sensemaking is a motivated, continuous effort to understand connections in order to anticipate their trajectories and act effectively"[27].

Based on the discussion in [28] and [58], Soo Yi et al. describe the process of sensemaking. Sensemaking is a:

- **Cyclic and iterative proceduce**: consisting out of a generation loop searching for representations, a data coverage loop instantiating the representations and finally shift representations;
- **Creation procedure**: being more about reasoning than discovery;
- **Retrospective procedure**: as people construct a framework and assign relevant information to a place withing this framework. If the data fits the framework well, the framework is confirmed, otherwise it may be updated or discarded;

An important remark made in [27] is that data fusion algorithms can reduce information overload, but they also pose challenges to sensemaking if the human can't form an accurate mental model of the machine, to understand why and how the algorithms are doing what they are doing.

The data and dimensionality reduction algorithms used in *SoundSuggest*, increase the amount of implicit information that needs to be interpreted by the end user[19, 57]. In order gain insight into the recommendation process, it is important that certain contextual information is retained. The contextual information we want to convey is two-fold:

1. The strength of the links between a recommendation and the user's profile;
2. The position of the user in his/her neighbourhood and the relation with those neighbours.

The first type of information is contained in parallel edges between items. For the second type of information, the active user's neighbours should be included in the visualization in one way or the other. In the resulting visualization the user's top neighbours are listed next to the graph. By hovering or clicking one of the listed neighbours, the relevant parts of the graph, i.e., items owned by the neighbour and the edges between them, are highlighted.

Processes of insight gaining

Although sensemaking can play an important part in gaining insight, it is not the only path to arrive at insight [58]. Yi et al. [58] identify four processes, that are often intertwined, through which insight is established:

- **Provide overview**: *In this process the individual gains understanding of the big picture of a dataset of interest. It allows the user to make a distinction between what is known to him/her and what is not.* The user gets an overview of the recommendation rationale.
- **Adjust**: In this process a person will explore a dataset by adjusting the level of abstraction and/or the range of selection. Typical actions involve filtering and grouping of data.
- **Detect pattern**: *In this process the user will try to identify specific distributions, trends, frequencies, outliers or structure in the dataset.* Through numerous interactions, the user is able to identify a pattern in the relationshpis between recommendations.
- **Match mental model**: In this process the gap between data and cognitive model is bridged, reducing cognitive load and linking the present visual information with real-world knowledge.

The link with sensemaking is found in the cyclic and iterative nature of sensemaking - provide overview, adjust and detected pattern can be applied iteratively, as well as its creative and retrospective aspects - adjust and detect pattern create hypotheses and test them through various interaction techniques[58].

It would be interesting to see if we can identify these steps in the user tests performed in this thesis as well.

Improving insight

Yi et al. [58] identify several ways in which the insight gaining process can be made more efficient. They list the system's interactivity, the quality of visual encodings and usability among others, as possible enablers for increased insight gaining. Naturally, improvident designs will act as barriers rather than enablers in the insight gaining process.

Interactivity of the system promotes the user's engagement into the dataset. Spending more time with the data will allow users to form more detailed and accurate hypotheses, and as a result greater insight[58]. At the same time, while using the visualization, the user will become more skilled at a task over time. Nonetheless, bare in mind that when performing long and tedious search tasks, vigilance will become an important aspect as well in the efficiency of data exploration[57]. Visual explanation systems may involve interaction techniques. For example, in the *SoundSuggest* application, users can click an item node. Related item nodes will then be highlighted with it as well.

Similarly visual encodings that are counter-intuitive will also increase the cognitive load. Other barriers on insight gaining are clutter, occlusion and data overload[58]. In the application we have built, we applied clutter reduction techniques for graph visualization.

Usability is another aspect that may have an impact on the insight gaining process, as controls that are hard to use will inevitably occupy some of the cognitive capacity of the user[58]. In the ISO standard ISO 9241-11, usability is defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"[54].

Note that usability should not be considered a one-dimensional property of a user interface. Nielsen identifies several characteristics of usability in applications[37]:

- **Learnability**: if the system is easy to learn, the user can get started quickly;
- **Efficiency**: if the system is efficient to use, it will be possible to complete more work in less time;
- **Error rate and severity**: if the system should be robust and minimize faults;
- **Memorability**: once the system is learned, acquired skills should not be forgotten easily;
- **Satisfaction**: the system should be pleasant to use.

Note that an application may not necessarily have to have a high value for each of these properties. An expert visual explanation system that is highly complex may score low on learnability, while still remaining a very usable system. On the other hand, a system for casual users may require high learnability, otherwise users will be reluctant to use it. The application we have built is aimed at users that want to dig deeper into the recommendation data to gain a better understanding of the recommendation system's rationale, but without going into high detail.

One of the aspects of the user tests we conducted involved testing the perceived usability of the application through test questionnaires. Through the think aloud protocol described ealier, some usability problems could also be detected through observation.

2.3.2 Interactive visualization

The relation between insight gaining and data visualization has been pointed out in other research. Colin Ware [57] describes interactive visualization as an "internal interface between the user and the computer in a problem solving system". Keim [25] notes that "idea behind visual data exploration, is to present data in a visual form, allowing the user to gain insight into the data, draw conclusions, and directly interact with the data".

In a chapter on visualization in [47], Tamara Munzner describes visualization as follows: "visualization allows the user to offload cognition to the perceptual system, using graphical data representations as a form of external memory. Therefore, by augmenting human capabilities, the data analyst is aided to understand, explore and form hypotheses of the data"[47]. In conclusion, the *visual data exploration* process can then be understood as a hypothesis generation process[25].

A reoccurring theme in visualization design is Schneiderman's mantra: "overview first, zoom and filter, and details on demand"[25, 47, 57]. First, the user looks for patterns of interest in the data space. Next the user focuses on one or more of these patterns, and starts looking at the data on a more detailed level. The user can then draw his/her conclusions and explore the data space further[25].

In what follows, we try to describe how a human interacts with interactive visualization on a cognitive level. It will be clear that some parallels can be drawn with the insight gaining process. This should come as no surprise, since these processes are intertwined[25, 57, 58].

In [57], interactive visualization is characterized by three classes of feedback loops:

- **Data selection and manipulation loop**: the user selects and moves objects that are selected through simple interactions based on eye-hand coordination;
- **Exploration and manipulation loop**: the user tries to find his/her way through a large visual data space;
- **Problem solving loop**: the user forms hypotheses about the data and refines them through an augmented visualization process.

The next subsections describe each feedback loop in greater detail.

Data selection and manipulation loop

In the data selection and manipulation loop a user will try to interact with the visualization. For example by clicking a node in a graph, hovering a list of elements and so on. The quality of performance of selecting and manipulating data on a screen, depends on certain factors. Colin Ware discusses the following attributes:

- **Reaction time**: This is the amount of time for a user to identify and select certain objects.
- **Types of interaction**: different types of interaction will have a different influence on user performance. In our visualization only two types of interaction are supported: clicking and hovering buttons, links or words.
- **Learning**: The speed at which a user performs a task may decrease over time, as the user becomes more skilled at executing the task.

Each of these attributes may be influenced by different factors. Through experiments predictions for these parameters have been captured in various laws such as Hick-Hyman law (reaction time), Fitt's law (selecting an object in a two-dimensional space) and the power law of practice (learning effects)[57]. Based on these laws, it is expected that reaction time will decrease over time as users become more familiar with the visualization.

Exploration and navigation loop

In the second loop the user navigates through the data space. The basic navigation control loop is described as an iterative process that involves two distinct aspects[57]:

- **Human**: The user gains understanding of, i.e., gains insight into, the data space through a logical, spatial model. Parts of this model may be encoded in the longterm memory, on the condition that the data space is maintained for a long enough period of time.

- **Computer**: The visualization may be updated and refined based on user input. Through clicks and hover queries the user will be able to change parts of the visualization.

When exploring spatial maps, a user is confronted with the *focus-context problem*, i.e., "the problem of finding detail in a larger context"[57]. The objective is to see the relation between the larger context and the details, rather than finding details. Ware goes on to discern between spatial, structural and temporal scales in which the focus-context problem manifests itself[57].

Ware and Mitchell remark that the human visual system is already well-adapted to the spatial focus-context problem[57]. Therefore, they argue that when designing a display, it should already try to take a maximal advantage of these perceptual skills. This can be done by displaying as much data as possible, without causing visual overload[57]. Of course, there are always computational costs, either finding the best way to represent large amounts of data, or instead, using interaction techniques to delve deeper into the data[19, 47].

Problem solving loop

The problem solving loop can be described through means of a *visual thinking algorithm*[57]. Such an algorithm combines perceptual and cognitive actions into a process, as the user interacts with the visualization and explores the data space. As we want to keep the *cost of knowledge* low, it is obvious that the cost and time complexity of each of these actions should be kept at a minimum. The cognitive system that runs these algorithms, is made up out of several different components[57].

Visual queries translate a hypothesis into a cognitive task. The result of a visual query can be a pattern or lack of a pattern. To support vsiual quaeries, actions that support information search are executed, called *epistemic actions*. Out of all epistemic actions, eye movements have the lowest cost, before mouse selection and hover queries. At the lowest level, elementary features such as color, texture information, and local edges are extracted from the image. Next, patterns are recognized by combining these features. Through eye movements possibly interesting patterns are explored. In the visual working memory, which forms the intermediary between the long-term memory and the incoming patterns, patterns are processed as *object files*; these are a combination of visual attributes and semantic meaning. Internal images can be combined with external images to construct and test hypotheses about the visualized data[57].

We will try to approximate a visual thinking algorithm applied by users when exploring SoundSuggest's visualization. The algorithm in table 2.1 is the degree-of-relevance highlighting algorithm derived by Ware and Mitchell [57].

Based on this algorithm, we can make an estimation of the efficiency by which the user may to use the visualization. We can combine this with the insight gaining process, described in section 2.3.1, to get a better understanding of the user.

2.4 Visual explanation systems

In this section we will take a look at visual explanation systems that already exist and that have been evaliated as well. Five different applications are discussed: *PeerChooser*,

Display environment: A display containing many symbols representing entities linked by a complex set of relationships.

1. *Construct a visual query to find a symbol that may lead to useful information (information scent).* For example, the user wants to find out more about a particular recommendation, shown in the visualization.

2. *Execute an epistemic action by selecting a symbol.* A symbol corresponds to either a user from the user list, or node on the graph in the visualization.

3. *Computer highlights all symbols with a high degree of relevance to the selected symbol.* These are relevant parts of the graph, i.e., items owned by the neighbour and the edges between them, are highlighted.

4. *Execute a visual pattern query among the highlighted symbols for additional information scent.*

5. *If a very high relevance symbol is found, execute an epistemic action to drill down for additional information. Usually this will be presented in a different display window.* In this window artist and user metadata are shown, for example artist playcount, shared top artists et cetera.

6. *Repeat from step 1 as needed, cognitively marking visited symbols.*

Table 2.1: Degree-of-relevance highlighting visual thinking algorithm by Ware and Mitchell [57].

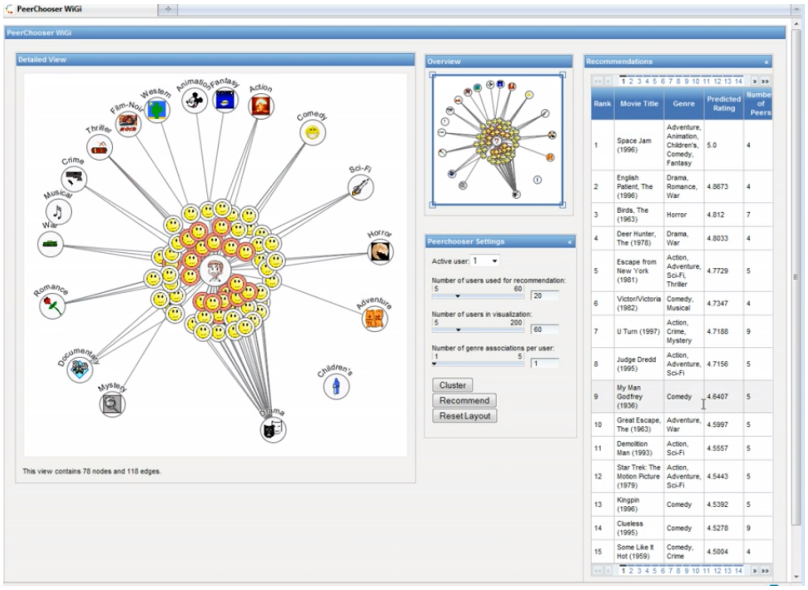


Figure 2.9: The PeerChooser interface.

Pharos, *SFVis*, *SmallWorlds*, and *TasteWeights*.

Next a comparison is made between these systems, based on a number of objectives for explanation systems listed by Tintarev and Masthoff in [51].

2.4.1 PeerChooser

PeerChooser is a "collaborative movie recommender system with an interactive graphical explanation interface"[41]. It aims to address the black box problem, described in section 2.1.3[41].

The application shows a peer graph of the user's neighbourhood. The visualization uses a force-directed graph layout where the on-screen distance between nodes corresponds an approximation of the node similarity. The active user is able to manipulate this neighbourhood by repositioning nodes of the graph. To deal with the high dimensionality of the data, extra cluster nodes are added to the vsiualization. These cluster peer nodes by genre[41].

By moving a cluster node closer towards the active user's node, all user nodes associated with this cluster will be drawn closer towards the active user node. As a result, these profiles will be temorary considered more similar than before. Similarly individual users can be moved closer towards or further away from the active user node[41].

Figure 2.9 shows the graph-based interface of the PeerChooser application.

2.4.2 Pharos

Content-centric social websites, such as blogs and discussion forums, contain vast amounts of fast growing information. Recommendation systems have been developed to help users find the information they are looking for. The *Pharos* application tries to address two distinctive problems that present themselves in this context: the cold start problem and the black box problem[59], as described earlier in section 2.1.3.

Figure 2.10: The Pharos social map. Colours indicate activity within a certain group.

As they hope to overcome previously defined problems, Zhao et al. [59] collect and visualize content-related social behaviour. The resulting data set is tranformed into a social map. The social map provides a context for new users, addressing the cold start problem. Secondly, the user can explore the social map to increase understanding and user interaction, in an effort to overcome the black box problem.

The generation of the social map takes place through the following three step process:

- **Community extraction**: a map depicting 'which users are talking about what'. Starting from either relationships, people or content communities can be derived;
- **Community/item/people ranking**: the next step is to rank these communities. The 'hotness' can be measured on content, people authorities and so on;
- **Community labeling**: describing what each community is about.

An example of what the resulting visualization looks like, is depicted in figure 2.10.

2.4.3 SFVis

SFVis (Social Friends Visualization) is an application that helps users explore and find friends interactively under a context of interest. The system is a hybrid approach of social tags and social networks. The SFVis framework transforms a data model (social tags and social networks) into a visual form. Users can both manipulate the input and visuals on demand.

Social tags can form a network. Within this structure clusters may arise. From this cluster tag network a hierarchy is derived. A compound graph is generated from the tag hierarchy and social networks.

A mapping function assigns actors in a social network to a tag tree. The actor similarity algorithm in SFVis considers both structure similarity in a social network and semantic similarity in a tag network. These scores will allow the recommendation system to compute friend suggestions.

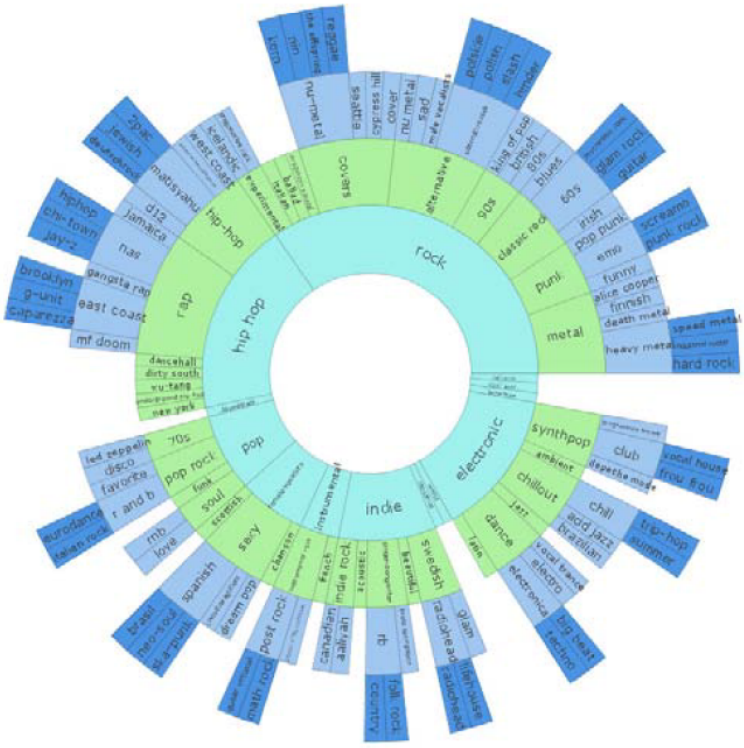


Figure 2.11: SFViz graphical user interface: tag tree.

SFVis uses circular visualizations for the different trees and graphs for both views as well as interaction with the user.

2.4.4 Smallworlds

In [16], Gretarsson et al. used the Facebook API to create an application to generate social recommendations for Facebook users. Unfortunately the Facebook API does not support unauthorized reading of item preference information beyond the immediate friend group. This would not have been a problem unless traditional collaborative filtering strategies tend to produce item suggestions of inferior quality for small items. In this case however the research team relies on the social filtering through the active user's peer group[16].

SmallWorlds is "a visual interactive graph-based interface that allows users to specify, refine and build item-preference profiles"[16]. The system promotes transparancy in the recommendation process, and gives the user a sense of control over the recommendation process through interactions. This way, Gretarsson et al. try to further overcome the limitations of their recommender system[16].

SmallWorlds uses a five-layered design to create suggestions:

1. the active user's node;
2. the active user's profile items;
3. friends who have items in common with the active user;
4. items that are not in the active users profile, but are liked by friends in layer 3;

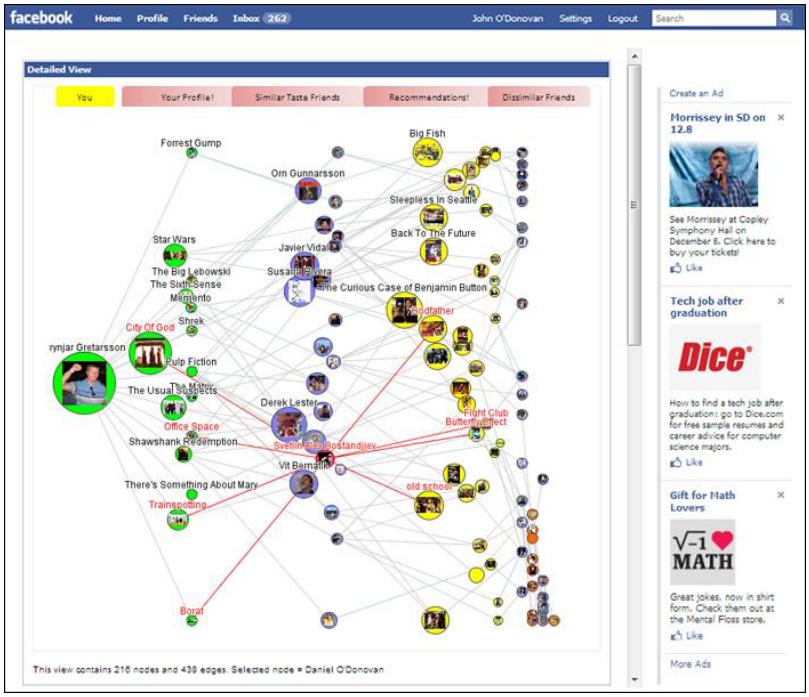


Figure 2.12: The SmallWords interface.

5. friends who have no items in common with the active user and items in their profiles, but not items in the profiles of friends in layer 3.

The user can move nodes in each layer further or closer towards the active user's node to adjust the weights of each node. This is used in combination with similarity functions to calculate the suggestions. Figure 2.12 shows a screenshot of the application.

2.4.5 TasteWeights

TasteWeights is a hybrid recommender with an interactive graphical user interface[2]. The application allows the user to express his/her preferences changing the weights of incoming data sources.

One of the challenges Bostandjiev et al. try to address is that "social web APIs and other data sources are constantly evolving, and traditional recommender system techniques such as automated collaborative filtering need to adapt to the changing environment of the social web"[2] (cf. Smallworlds). They introduce two enhancements for the traditional techniques. Multiple web sources, namely from *Facebook*[5], *Twitter*[6] and *Wikipedia*[7] are combined when computing the recommendation. This combination provides a hybrid of different recommendation strategies, namely: collaborative filtering, expert-based and content-based respectively. The second enhancement is a new user interface that provides transparency into the recommendation process.

There are three levels to be distinguished that are represented visually as well:

[5] `https://www.facebook.com/`
[6] `https://twitter.com/`
[7] `http://www.wikipedia.org/`

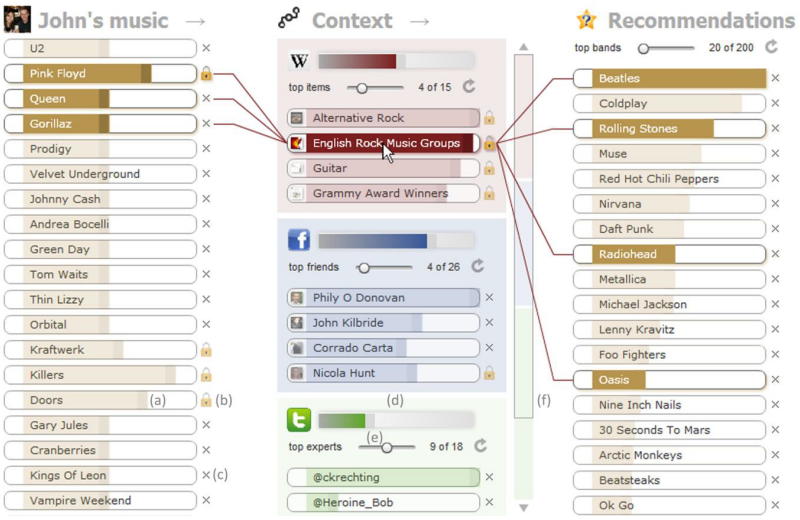


Figure 2.13: The TasteWeights interface.

- **Profile layer**: liked items on *Facebook*;
- **Context layer**: items coming from different sources, namely *Twitter*, *Facebook*, and *Wikipedia*;
- **Recommendation layer** containing the actual recommendations.

Figure 2.13 shows the corresponding visual representation of each of these levels. Edges connect relevant parts between each of these levels on the visualization, in an attempt to explain the provenance of item recommendations. The user can influence the outcome displayed in the recommendation layer by attributing weights to the nodes in the profile layer and context layer[2].

2.4.6 Comparing visual explanation systems for item recommendation

The following listing is adapted from [51]:

- **Transparency**: explain how the system works;
- **Scrutability**: allow users to tell;
- **Trust**: increase users' confidence in the system;
- **Effectiveness**: help users make good decisions;
- **Persuasiveness**: convince users to try or buy;
- **Efficiency**: help users make decisions faster;
- **Satisfaction**: increase the ease of usability or enjoyment.

Table 2.2 gives an overview of the performance of each visual explanation system based on the criteria listed by Tintarev and Masthoff in [51].

Table 2.2: A comparison of the visual explanation systems, based on the criteria by Tintarev and Masthoff listed in [51].

	Transparency	Scrutability	Trust	Effectiveness	Persuasiveness	Efficiency	Satisfaction
PeerChooser	x						
Pharos							
SFVis	x						
SmallWorlds							
TasteWeights	x						

2.5 Summary

First recommender systems were discussed: we looked at types of recommendation algorithms, a number of properties of recommender systems, and finally some challenges of recommender systems. One of these challenges was the black box problem. We saw that visualizations can be used to develop a white box model.

Next, aspects of information visualization were discussed: we look at different types of data and visual encodings. We saw that the recommendation rationale behind collaborative filtering could be interpreted as a set of relationships between users and items. As a result, a graph-based approach could be used to visualize this data. Using data, dimensionality, and clutter reduction techniques along with interaction techniques, we developed a visualization that could serve as a white box model for collaborative filtering.

Then we described how users can gain insight into interactive visualization. We look at evaluation techniques for insight gaining. Finally a visual thinking algorithm was proposed that describes how a user could gain insight into the visualization developed in the previous section.

In the last section we looked at five visual explanation systems that already exist. A comparison of these systems was presented based on the evaluation criteria by Tintarev and Masthoff listed in [51].

The next chapter describes SoundSuggest's target audience, the application's story board, and use cases. Chapter 4 gives an overview of the evaluation and design iterations. Chapter 5 describes the implementation of the SoundSuggest application. In chapter 6 conclusions and future work are discussed.

Chapter 3

Requirement analysis

3.1 User profile

The target audience of the application includes users that look for new music or artists based on generated recommendations. Table 3.1 tries to establish a profile of the target users. Note that most of this user profile is what we expect the application's users to be like, rather than the result of surveys or other types of investigation.

Table 3.1: User profile 1: sketching the targeted audience

Skill set:	Has basic knowledge of computers; Uses mouse for navigation; Uses keyboard for entering text; Is familiar with traditional website layouts; Understands English;
Behaviour:	Pays regular visits to sites like or similar to last.fm, imdb.com, netflix.com, youtube.com, and amazon.com and has an account on one or more of these websites; Has a Facebook or other social media account; Uses applications such as iTunes, Windows Media Player, and Spotify to listen to and purchase music; Uses recommender systems to find new music, movies, books, et cetera.
Interests:	Music, videos, and other kinds of multimedia. Online social networking.
Demography:	Aged between 18 and 30 years old; Both male and female users;

User goals with a relevant a part of the application's functionality are the following:

- The user wants suggestions, filtering out possibly interesting items from the vast item space. *Suggestions are listed by the system, based on the user's interests. The user can add suggestions to his/her profile.*

Figure 3.1: The story board for the SoundSuggest application.

- The user wants to gain insight into the reasoning behind the suggestions. *Through the explanation system, the underlying conceptual model is visualized.*
- The user wants an indication of how reliable the suggestion is. *By providing contextual information for each recommendation, the user can estimate how well the recommendation corresponds to his/her profile.*

3.2 Story board

The story board of the application is shown in figure 3.1.

3.3 User story

Imagine you have a music library with a number of tracks in it. No doubt you will like certain tracks more than others. At a certain point you will want to expand your library. It is only natural that you will want to add music that is similar to the music you already like, but where should you begin to look for this kind of music? Over the last decade, systems have been developed to compute what tracks, or in the more general case, items or information, would be of interest to you based on your listening history and/or track ratings. These kind of systems are called recommender systems.

Let us assume you have plugged some recommender system into your music library and you have received a list of music suggestions. Which of these recommendations should you choose? Of course you could go through them all one by one, but that might take up quite some time. What it comes down to is that you don't know how the recommender system computed these recommendations, and as a result, you have a hard time making an educated decision where to start.

Let's say that you have installed the the recommender system with an integrated explanation system. The explanation system visualizes how the items in your library are related to the recommendations, and provides additional statistics. Now, finding new, interesting music will hopefully become easier than ever before.

3.4 Use case diagram

Based on the discussion in section 2.3.1, four interactions can be identified: hovering of items, hovering of users, clicking of items, and clicking of users. The use case diagram is presented in Figure 3.2 lists each of these interactions. Tables A.1, A.2, A.3, and A.4 in appendix A describe each use case in greater detail.

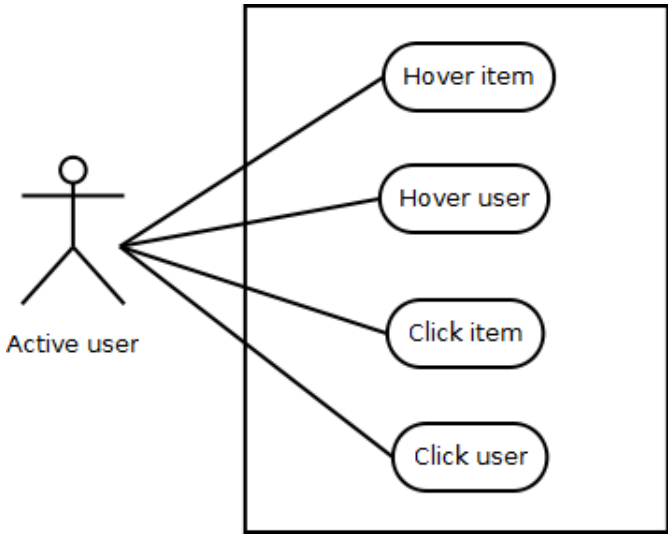


Figure 3.2: Use case diagram of the SoundSuggest application.

Chapter 4

Iterative development

This chapter describes how the idea presented in chapter 2 is tested and improved through different development cycles. First the evaluation and development techniques used for the tests are described.

4.1 Methodology

4.1.1 Prototyping

There are three types of prototypes that were used in the iterations:

- Paper prototype;
- Digital prototype with 'fake' data or interaction effects;
- Digital prototype with working implementation.

It is obvious that for each category the resources that are required to build the prototype differ. The objective is to filter out most of the issues in the low cost designs, avoiding a greater cost in the more expensive prototypes.

Paper prototyping is defined as "a variation of usability testing where representative users perform realistic tasks by interacting with a paper version of the interface that is manipulated by a person 'playing computer', who doesnt explain how the interface is intended to work"[49]. It is a technique for designing, testing, and refining user interfaces [48], and is closely related to usability testing [49]. In the last decade it has become a regularly applied technique in major businesses such as IBM, Digital, Honeywell, and Microsoft among others[48].

Digital prototypes capture a portion of the functionality in an application. In the early stages of the design process, this application usually works with static data, or a limited number of screen transitions. This allows for more flexibility when certain functionality has to be altered. In later iterations the static data is replaced with the real data. In such prototypes, the effects of performance of algorithms and interface responsiveness can be analyzed in greater detail.

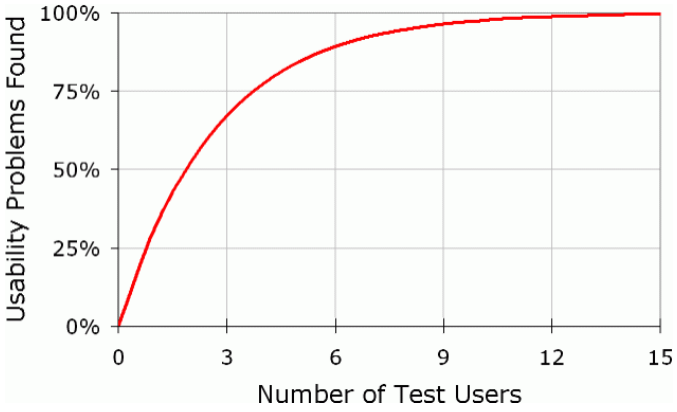


Figure 4.1: The curve shows the user test's diminishing returns beyond a certain amount of test users; adapted from `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`.

4.1.2 Evaluation techniques

The evaluation of an application prototype can be performed using one or more different techniques, and based on a range of varying criteria, such as: usability, usefulness, meaning, efficiency, accuracy and so on. Various techniques exist, such as questionnaires, usability engineering, expert evaluation, and usage tracking[7].

Methods may be *formative* or *summative*. Formative means that the evaluation occurs simultaneously with user task execution. Summative occurs after the user has performed all the required tasks[7].

The goal of the evaluation is at one hand usability testing, and at the other hand verifying whether or not the visualization can produce insight by test users as described in section 2.3.1. Both objectives can be tested through a variation on *usability engineering* called *think aloud* user tests. Additionally, a type of questionnaires called *system usability scale* (SUS) *questionnaires* are used to obtain a quantification of the perceived usability by test users. Such a quantification may allow us to identify positive or negative trends in the usability throughout the iterations.

In order to perform reliable usability tests, the test users have to be representative for the actual user population[7, 40]. The tasks that are being used, have to be representative of the system usage. Tasks also have to correspond to research questions to obtain relevant results[48].

The number of users can often be limited. As the number of detectable problems is likely to be finite, from a certain point on testing more users will not produce new or better results[7, 39]. The graph in figure 4.1, adapted from [39], illustrates this phenomenon.

Nielsen argues that as a rule of tumb, five test users is enough to acquire reliable and valuable test results. Instead of doing one test with 15 users, use three iterations with 5 users each. Based on the graph, the first iteration will discover the majority of the usability problems; the next two tests will uncover the remaining 15% of issues. Of course, this only holds on the condition that tasks performed by the users are similar for each iteration. Between each iteration, corrections are applied to the design[39]. Between these groups of tests, detected usability problems are addressed, and hopefully resolved which can be verified in the next iteration.

Advantages	It is cheap to perform; It is robust; It is flexible; It is convincing; It is easy to learn;
Disadvantages	It creates an unnatural situation, as users usually don't say out loud everything they are about to do or think; The user may tend to filter his/her statements to avoid saying things that he/she may find silly or uninteresting; The facilitator may introduce bias in user behavior if he/she provides too much information when answering or instructing users;

Table 4.1: Advantages and disadvantages of the think aloud protocol.

Usability engineering

In [7], two methods are described to perform usability engineering tests: usability labs, and think aloud testing. In a usability lab the user is observed while performing certain tasks. Data on task completion time, mouse clicks, eye-movement can be collected. Direct observation or cameras can be used to observe the user. To mimic real-life situations, also complete settings can be recreated in which the users would normally use the application[7].

Using usability labs can be rather costly, since labs need to be available and the required equipment may be expensive. The think aloud protocol is a variation on the usability lab method and is cheaper to perform, cf. 'discount usability engineering'[7]. During a think aloud test, the user describes his/her reasoning for each action he/she undertakes[38]. This method has several advantages and disadvantages, as listed in table 4.1, based on [38] and [48].

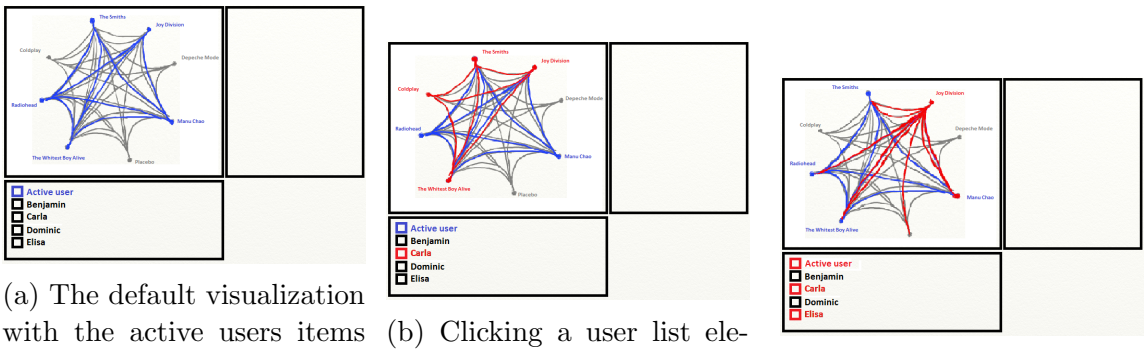
Questionnaires

To obtain information other than observational data, the user is presented with a summative questionnaire. Questionnaires are used to obtain subjective information from the user about the user's experiences. Table 4.2 lists several advantages and disadvantages of the use of questionnaires in usability studies, based on [26].

There are several standardized questionnaires. The one used for the application evaluation is the system usability scale (SUS). A system usability scale test is a questionnaire that consists out of ten specific questions that attempts to measure the user's perception of the application's usability. Each question is answered by checking one out of five checkboxes: checkbox one corresponds to strong disagreement with the statement, the fifth checkbox corresponds to strong agreement with the statement[45]. The ten questions are listed in appendix B.

Advantages	Evaluates the point of view of the user; Measures gained from a questionnaire are to a large extent, independent of the system, users, or tasks to which the questionnaire was applied; Quick and cost effective;
Disadvantages	Only the user's reaction as the user perceives the situation; Lack of detail, as questionnaires are usually designed to fit a number of different situations; Subjective data must be enhanced with performance, mental effort, and effectiveness data.

Table 4.2: Advantages and disadvantages of the questionnaires.



(a) The default visualization with the active users items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Figure 4.2: A selection of the screens used in the user study with paper prototype.

4.2 Iterations

4.2.1 Iteration 1: paper prototype

The prototype

The paper prototype that was tested consisted out of a single screen. On this screen a visualization was drawn by hand. A number of copies of this drawing and by changing certain parts of the visualization, transitions within the vsiualization could be mimicked as users interacted with the screen. A selection of these screens are shown in figure 4.2.

Test parameters

Five test users were selected that often listen to music. Although they did not necessarily use recommendation systems to actively search for new music, they had a vague impression of what these systems were. Test applicants were between 22 and 26 years of age.

The objective of the test is two-fold:

1. **Insight**: Verifying whether or not the user can gain insight into the recommendation rationale.

2. **Usability**: Finding out the perceived usability of the application? Discovering usability issues through observation?

The list of tasks that were performed by the users is listed in appendix C.1.

Test results

The first part of the test was aimed at forming a first mental model of the system without interacting with the visualization. When participants were asked to describe what they saw and try to explain the visualization, all of the participants identified the edges as certain relationship between artists. Most of them interpreted the relationship encoded in the edges as a content-based relationship; for example artists that have similar genres are connected.

Blue edges were usually correctly associated with the highlighted active user profile. This insight made one of the users see that the edges represent a co-occurance relationship, i.e., if a user has any set of two items in his/her profile, these items are connected.

If users became aware of the fact that blue nodes and edges corresponded to items that were already owned, item suggestions were easy to point out. In some cases this waited until the second step of the evaluation process where interaction was allowed.

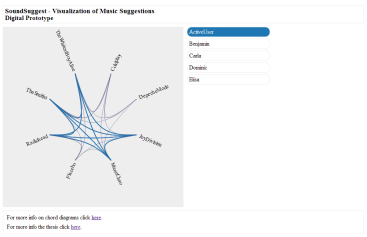
The test users were asked what kind of interactions were possible with the visualization, and what the effects of these actions would be. All of the users listed left mouse clicks. Also dragging and scrolling were suggested by some participants. In that case participants were simply told that this kind of interaction was not supported. They expected to be able to manipulate the edge trajectory by dragging the edges were they wanted. However, clicking an edges is also not supported by the visualization. The effect of clicking a node was usually correctly predicted, although some users did not immediately see that related users would be highlighted as well.

In order to avoid restricting the user to predefined action patterns, the user was relatively free to explore the visualization in the second part of the test. Most participants started by clicking another user's icon and noted the resulting highlights in red in the graph. For one user the tasks in this step were a mere confirmation of the already established mental model. For most users this turned out to be an important moment in adjusting the first model. When alternating between clicking users icons, as well as between artist nodes and artist nodes and user icons the other users were able to correct their model in this step to finally form the correct picture of what the visualization was trying to convey. For two users this took significantly more time than for the other two remaining users.

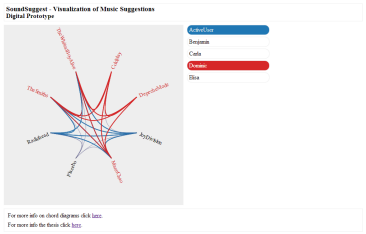
The understanding of the relationship encoded in the edges, is key to grasping the whole idea behind the visualization. Once this was understood, all users could explain the recommender rationale. Moreover, users were able to point out an item recommendation that was more favourable than another suggestion. For example using the total number of links to the active user profile, the total amount of related users, or a strong connection with a particular favourite item.

Conclusions

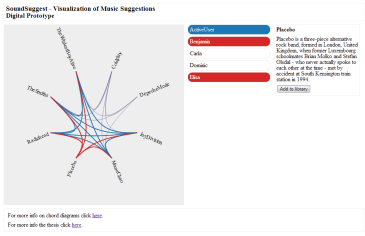
In conclusion, one user managed to get the complete mental model correct in the first step of the insight gaining process. The others were able to correct it in the second step. The model helped identifying a particular suggestion as more interesting than an others based on what they learned from the visualization.



(a) The default visualization with the active users items highlighted.



(b) Clicking a user list element.



(c) Clicking an item node.

Figure 4.3: A selection of the screens used in the user study with the first digital prototype.

4.2.2 Iteration 2: first digital prototype (SoundSuggest 1.x)

The prototype

As the test users were able to discover the recommender rationale using the visualization, and no notable usability had arisen, we started working on the digital prototype. This prototype used static data, but already supported all the use cases listed in appendix A. The resulting prototype can be seen in figure 4.3.

Test parameters

Test users were selected in a similar manner as in the first iteration. Two test users from the previous iteration were tested again, the other three were new test users. The test users that had been tested before, were tested first.

The objective are the same as in iteration one. However, in addition to these objectives we want to find out how successful the transformation from paper to digital prototype had been. Also feedback was asked on a help file that was made for the application.

The tasks remained the same as in the previous iteration. The test users that were tested in the first iteration also were asked if there were any improvements or new issues going from the paper prototype to a digital one. Remarks made by these persons were also presented to the other test users.

Test results

Test users listed the list of neighbours, artist names. Most of them also noted that certain items and edges were coloured blue. One user also immediatelt saw that some of the other users and items did not have this colour and explained that these items corresponded to items recommended by these users. This participant further clarified that he expected that for each neighbour a similar coloured structure would exist as for the highlighted active user. When asked how this helped him understanding the recommendation rationale, the test user explained that these were users that had similar tastes as the active user. Items in these profiles were then candidate recommendations.

The other users did not find the recommendation rationale in the first step. When interacting with visualization however, they were able to tell similar stories. Again, the key to understanding the visualization turns out to be understanding what contextual information is that is encoded in the link between artist nodes. For most users this could

be discovered by comparing the neighbour profile with the active user's profile on the graph.

With regard to the first iteration, one particular difficulty was mentioned by both users that tested the paper prototype: in the paper prototype parallel edges were easy to discern, but in the edge-bundling algorithm used, parallel edges will overlap in the resulting visualization. This makes it harder to see the connection between a clicked artist, and the number of neighbours that were highlighted. However, when the new test users were asked if they saw this as a problem, they acknowledged that this could help, but didn't see it as a major improvement. An improvement in the digital prototype was that the smoothness of interactions had of course greatly increased, which made it much easier to compare items and users, according to the test users.

Two users thought that deselecting an item or user should be triggered by clicking somewhere outside the graph as well.

When asked to add one of the recommendations to the active user profile, all users could give one or more reasons, similar to the first iteration.

The help files should have a table of contents or some sort of overview.

Conclusions

Overall the going from paper to digital prototype was successful, apart from the issue mentioned earlier. Although it would be interesting if this problem could be solved, users indicated that was not a particular barrier for gaining insight into the recommendation rationale.

Different colours for hovering and selecting, although others thought this wasn't really necessary. Users suggested to add an option to choose between different encodings with an additional legend for the meaning of the different colours. Also options to alter the number of items shown were considered useful additions to the application's functionality.

4.2.3 Iteration 3: second digital prototype (SoundSuggest 2.x)

The prototype

The layout from the previous iteration was retained, but this time the visualization was incorporated into a chrome suggestion that could be injected directly into the Last.fm recommendations page, using real data.

An option menu was added with options to alter the data settings, and colour encodings. Data settings that could be answered were the number of recommendations shown, the number of top artists from the active user's profile shown, and the number of neighbours included in the visualization. Another option was to change the threshold. The threshold corresponds to the clustering range of the data collection algorithm. A low threshold value will allow the user to have a link to a certain artist without owning it. Instead the user may own one or more related artists, i.e., the neighbour is required to have a link to an item inside a cluster of items, rather than a particular item.

To solve the problem of deselecting an item, an addional button was added in the menu bar.

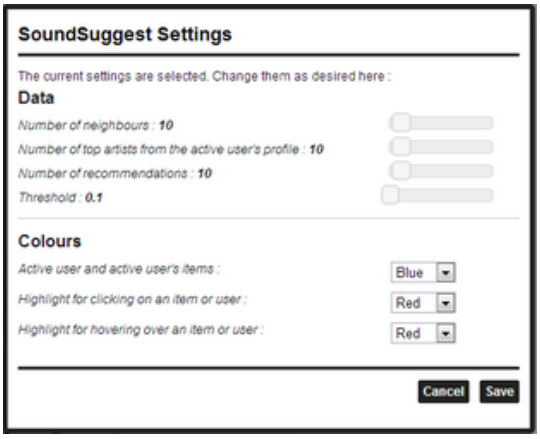


Figure 4.4: The settings menu of the second digital prototype.

Test parameters

Five test subjects were selected that were Last.fm users between the ages of 22 and 26. Three of them were regular users, the other two only used Last.fm occasionally. Two users had tested both the paper and first digital prototype, one other user already had tested the paper prototype.

The objectives of the test remained the same. In addion to the previous objectives the performance of the application was investigated as well. Concerning usability, there were three areas of interest:

1. **Visualization**: The general usability of the visualization according to new users.
2. **Option menu**: General usability of the options menu.
3. **Chrome extension**: The placement of the application into the Last.fm recommendations page.

The tasks listed in appendix C.2 were used to investigate these areas. Insight by new users could again be tested using the scheme in appendix C.1. To get an idea of the learnablity and memorability of the application, test users from previous iterations were asked to explain the visualization rationale again before the rest of the test.

Test results

When testing insight no notable differences from the previous iteration occurred. The new users also needed interaction with the vsiualization before the recommender rationale could be discovered.

Users from previous iterations were able to recall the visualization rationale from previous sessions, although two of them needed to interact with the visualization before they could remember it accurately.

The settings menu was found by all users when asked to change the number of visualized items and/or users. One of the remarks when changing the data settings, was that the visualization would take long to load. As there were now more edges and nodes,

some scalability issues came into play: some users complained that the increased number of edges would create clutter that made it hard to compare profiles.

Another issue that was mentioned was that it was hard to distinguish between recommendations once the test user started hovering over the listed neighbours. A test user from the previous session noted that this was less of a concern in the previous prototype, as the number of nodes much lower.

The threshold option turned out to be very confusing, even with an explanation from the help files. Also, the results of changing the threshold were not visually pleasing either. As soon as the threshold would be over 0.1, the connectivity dropped and some edges that were previously connected were no longer connected.

When adding a recommendation to the user profile, to see the changes in the profile, the whole page needed to be refreshed instead of just the visualization. Also, if the user would refresh or navigate away from the page, all of the data would have to be reloaded.

Conclusions

Apart from the threshold option, the settings menu posed no notable difficulties. It would probably be better to use a default setting for the threshold, and remove the option from the settings menu altogether. The other options can remain as they are.

If possible, it would be interesting to increase the data load speed.

4.2.4 Iteration 4: third digital prototype (SoundSuggest 3.x)

The prototype

To reduce issues with clutter, an additional option was added to the settings menu to alter the tension of the edges. This way, the user would be able to alter the layout to a certain extend.

To make it easier to distinguish between recommendations and top artists, the node labels for top artists are underlined in the graph.

A button to refresh the data and update the visualization was added to solve the problem of having to refresh the whole page to have the latest version of the data.

To avoid long waiting times when loading the page, the data that was loaded last was cashed. This way the latest data set could be loaded quickly from local storage. The refresh button can be used to get an up-to-date version of the data.

An example of the resulting visualization is shown in figure 4.5.

Test parameters

Ten test users were selected. Three of these users already had experience with the application based on the previous iteration. Two of the other test users had tested one the first digital prototype. The other test users were new users. All of the test users had some experience with Last.fm or other music recommenders like *Grooveshark*, *Spotify* or *Youtube*. If users didn't have a Last.fm account, they were asked to create one one to two weeks in advance and add their listening habits to their Last.fm profile.

For new users insight could again be tested using the tasks in appendix C.1. The tasks listed in appendix C.3 were used to test the following explanation system properties: *transparency*, *effectiveness*, *persuasion*, *trust*, and *satisfaction*.

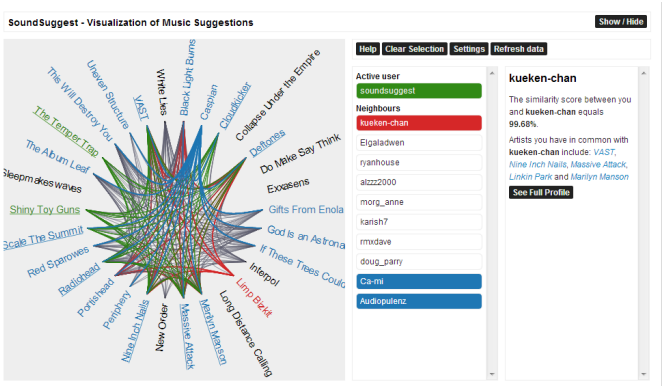


Figure 4.5: The third prototype of the SoundSuggest application. In this figure, the active user hovered an item while one of the users is selected.

Although *scrutanability* could have been tested by removing undesired recommendations from the list under the visualization, the visualization did not seem to include this information immediately when refreshing the data. It is also not really clear to what extend the information of removed suggestions is included into future recommendations by Last.fm's recommender algorithm.

Test results

Previously tested users liked the changes to the visualization. The tension parameter was visually pleasing. Most users liked a tension in the area of 0.45 to 0.60. Underlining owned artists made it easier to compare user profiles. One user indicated that simply sorting the artists by recommendation or top artist would make it even easier to distinguish between owned and recommended artists. The fact that the data set was cashed made users much more confident in clicking links, as they didn't had to worry about long loading times.

A problem that remained was the scalability of the graph. When visualizing a total of over 40 recommendations and top artists, loading times not only increase but the density caused by overlapping edges, makes it hard to compare user profiles.

All of the test users were able to describe the recommendation rationale. Most of them used a variation on the following: "Last.fm looks for users that have a similar taste based on the active user's favourite artists, i.e., neighbours. Last.fm decides which items owned by these neighbours are interesting for recommendation based on preference by neighbours (top artists) and the total number neighbours that own these items".

From the recommended items most users already knew more or less who these artists were. For some users this would increase trust, as they basically forgot about them when building up listening history. For other users this would actually decrease their trust in the system as they just were not interested in the recommendation. Interestingly Last.fm's own explanation for recommending the item was usually displeasing. Last.fm would justify the recommendation by listing a number of 'similar' artists. Unfortunately, for some categories of artists there exists some bias in the recommendations. For example musicians that have a solo project and also played in a series of different bands lets Last.fm consider these artists as similar. Another example is that artist recommendations for bands and musicians that operate in a country with a less international music scene

would be influenced by regional effects. Belgian bands would be considered similar just for beging Belgian to the point were all similar artist pages on the Last.fm website would be Belgian regardless of their genre. Users confirmed that this kind of bias reduces the trust in the recommender system. On the other hand, it was interesting to see that this bias could actually be detected in the visualization as neighbours usually did not have edges going from items in their profile to these 'biased' recommendations.

When users did not know a certain recommendation, most users were interested if the recommendation occurred in one of the neighbouring profiles. In this sense the visualization helped persuade the users check out certain artists. From the six users that checked out an item they did not already know, four of them found at least one item that they liked and added to their profile. When doscovering a new item that they liked, users admitted this significantly increased their trust in the recommender system, even more than when they found an item that they liked but already knew about.

Conclusions

One issue that was resolved between the user tests was that the artist names that were used to create CSS `id` and `class` names sometimes contained special characters causing the visualization to not function correctly anymore. This was solved by using the hash value of the artist name instead. Although no new iteration was started for this alteration, this may have influenced some of the test results. Overall this only affected two participants.

By letting users choosing the tension parameter themselves, a better default value for the tension would probably be around 0.55.

In future iterations the problems with overlapping edges should be addressed, for example by decreasing the alpha value of egdes that are not relevant to the selection.

Overall, the application scores well based on the user's feedback and it enables users to discover some characteristics of Last.fm's recommender.

Chapter 5

Implementation: the SoundSuggest application

The application that was built for this thesis is called *SoundSuggest*. It is a chrome extension that uses the D3 JavaScript library and the Last.fm API to inject the explanation system into the recommendations page of Last.fm[1]. In this chapter we will discuss the technologies we have used to create the application, the software design of the application and some specifics about the implementation of the application.

5.1 Technologies

5.1.1 Chrome extensions

Chrome Extensions are applications written in *HTML*, *JavaScript* and *CSS*, that enhance to functionality of the Google Chrome web browser[13].

There are different types of extensions. Browser actions are applications that can be launched regardless of the web page you are at. They appear as a button with a specified logo in the toolbar of the Chrome browser. By clicking the browser action you can specify to open up a tooltip or a popup[11]. Page action extensions are meant to be shown when browsing specific web pages. They appear as an icon in the address bar. Page actions use content scripts to inject code into the web page[14].

The file `manifest.JSON` is one of the key areas of a chrome extension. It specifies the name and the version of your application as well as other important settings such as the type of the extension, scripts and security policies[12].

Many extensions use a two-layered structure in which you have a background page and UI pages or content scripts[14]. In the usual case the views are stateless and background pages are not. When the view needs some state, it requests the state from the background page. When the background page notices a state change, the background page tells the views to update[10]. Background pages can either be persistent or not. In the last case we are talking about so called event pages; they are opened and closed as needed[14].

There are various ways to use UI pages: you can open an HTML page in a popup, another tab or options page. The HTML pages inside an extension have complete access to each other's DOMs, and they can invoke functions on each other[14].

[1] `http://www.last.fm/home/recs`

Content scripts are JavaScript scripts that are used to interact with a webpage opened in a browser tab. An important remark is that you should consider a content script part of the webpage it is injected into, rather than its parent extension. It can modify the DOM of the webpage but not the DOM of its background page. However it can ask its background page for data through listeners in the background page's script[14].

5.1.2 The Last.fm API

The *Last.fm API*[2] offers great functionality such as the recommender system, Last.fm scrobbling and accessing and modifying your Last.fm profile information, aside from providing a large amount of data. To use the API, libraries have been developed for several technologies, such as *JavaScript*, *PHP*, *Python* and *Actionscript* among others[32].

To build an application using the Last.fm API, you have to create an API account first at `http://www.last.fm/api/account/create`. Once you have been registered, you will receive an API key and an API secret.

For testing purposes it will also be handy to have a Last.fm account of your own. So if you haven't got one already sign up at their website. You might also want to one or more of their *Scrobbler* applications. This will collect data from your music players to generate profile information that will be used to generate recommendations[31].

There are already several interesting applications that make use of the Last.fm API. Even more interesting perhaps is that some developers distribute free JavaScript libraries that act like a facade on the Last.fm API. The JavaScript library we will be using here, can be found on *GitHub*[3] and is written by *Felix Bruns*.

5.1.3 D3.js JavaScript Library

Visualizations for web applications can be built using *scalable vector graphics (SVG)*. SVG is an XML-based language to describe two-dimensional graphics[56]. It is supported by most of the latest versions of most popular browsers, including *Chrome*, *Firefox*, *Internet Explorer 9*, *Opera* and *Safari*[36, 55].

D3.js is a JavaScript library that uses the W3C standards *HTML*, *SVG* and *CSS* to build data-driven documents[3]. There are various tutorials explaining the basics on how to use this library.

In short, to get started the library should be included in your web page. Next, using the D3 selectors, elements can be added and removed easily from the web page. The library also offers a number of built-in algorithms, as well as a series of example visualizations that can be customized as desired.

5.1.4 Additional libraries

In addition to the Last.fm API JavaScript library and D3.js, four other JavaScript libraries were used, namely:

- **jQuery**[4]: "jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation,

[2] `http://www.last.fm/api`
[3] `https://github.com/fxb/javascript-last.fm-api`
[4] `http://jquery.com/`

and Ajax much simpler with an easy-to-use API that works across a multitude of browsers"[23].

- **jQuery UI**[5]: "jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library"[24].
- **Purl.js**[6]: a library built on the jQuery library to retrieve GET parameters from the web page's URL.
- **Spinner.js**[7]: a library that creates a spinner element with given parameters for customization.

5.2 Software design and application architecture

The architecture of the application is shown in figure 5.1. Five distict components can be identified that are of importance for the application:

- **Last.fm recommendations page**: The HTML will we injected into this page. Although it is not a part of the source code, it poses certain limitations on the script. For example one should be careful not to override certain CSS definitions, and the injected code should fit into the page layout to achieve better looking results.
- **Content script**: The content script creates the injected HTML elements, handles user input, and delegates calls to the Last.fm API to the background script.
- **Background script**: the background script deals with local storage and calls to the Last.fm API.
- **Local storage**: The local storage of the Chrome browser can be used to store preferences.
- **Last.fm API**: The Last.fm API handles calls and returns the requested content.

Figures 5.2 and 5.3 show the sequence diagrams of what happens when loading the application. The first time the application is loaded, the user will have to authenticate the application. If the user does this, the content script will retrieve the token from the callback URL, and get a session key from the Last.fm API. This session key then stored. When the application is started again, the stored key can be retrieved from the local storage. Similarly other settings are loaded from local storage. If none have been stored so far, the default settings are returned and stored.

Algorithm 1 shows how the data structure is constructed from calls to the Last.fm API. The resulting data structure is an approximation of the utility matrix on a local scale, i.e., the neighbourhood of the active user and the top artists of the active user. The time complexity of the algorithm depends on the number of artists, i.e., the number

[5]

[6] `https://github.com/allmarkedup/jQuery-URL-Parser`

[7] `http://fgnass.github.io/spin.js/`

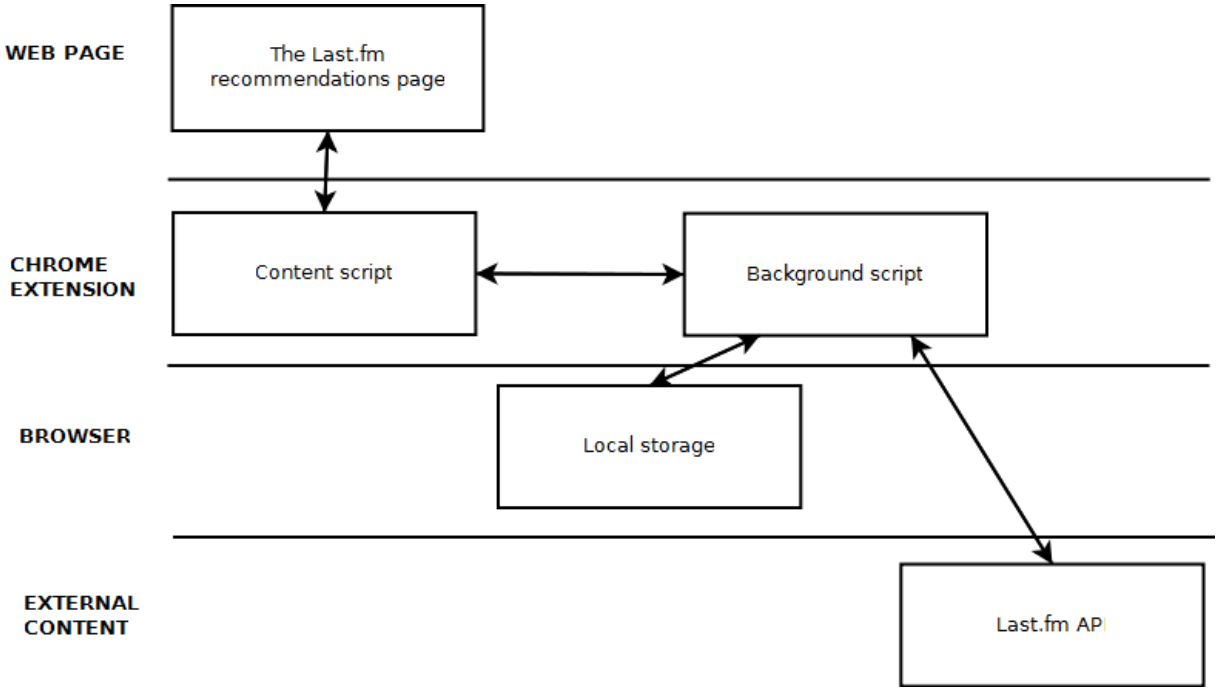


Figure 5.1: The architecture of the application.

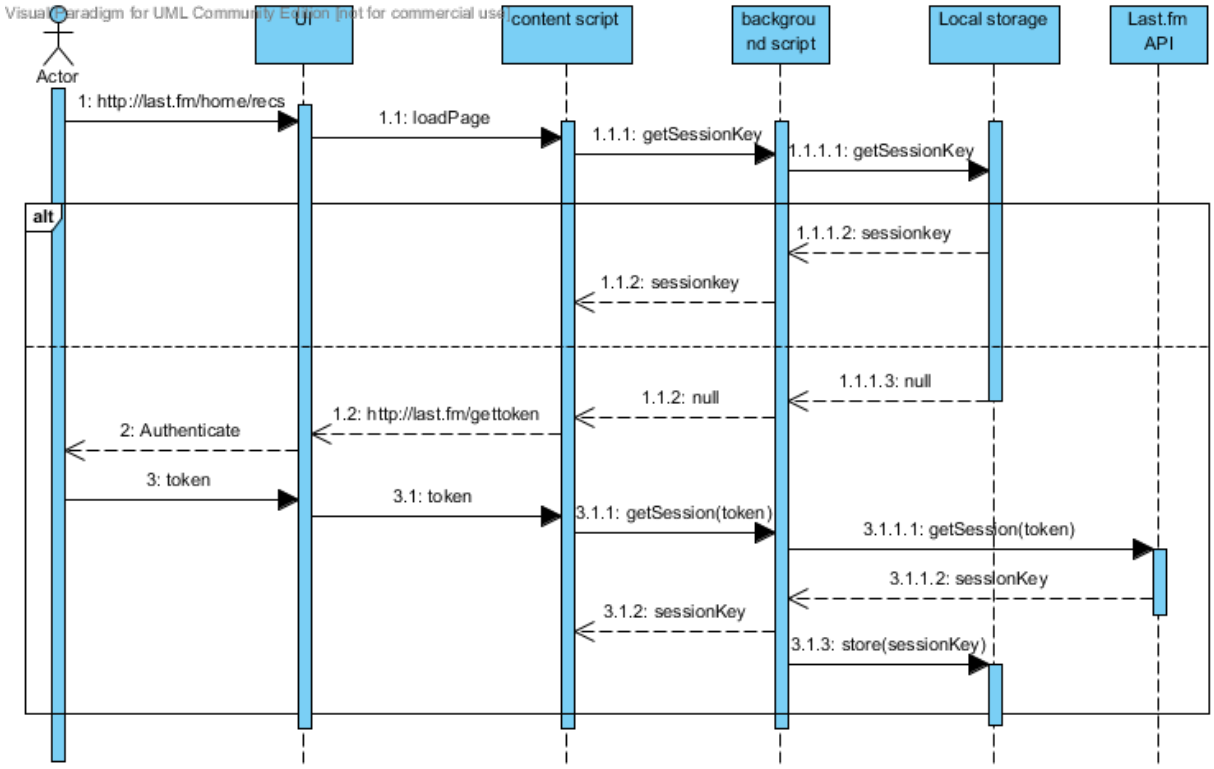


Figure 5.2: Sequence diagram: opening the Last.fm recommendations page part 1: retrieving a session key.

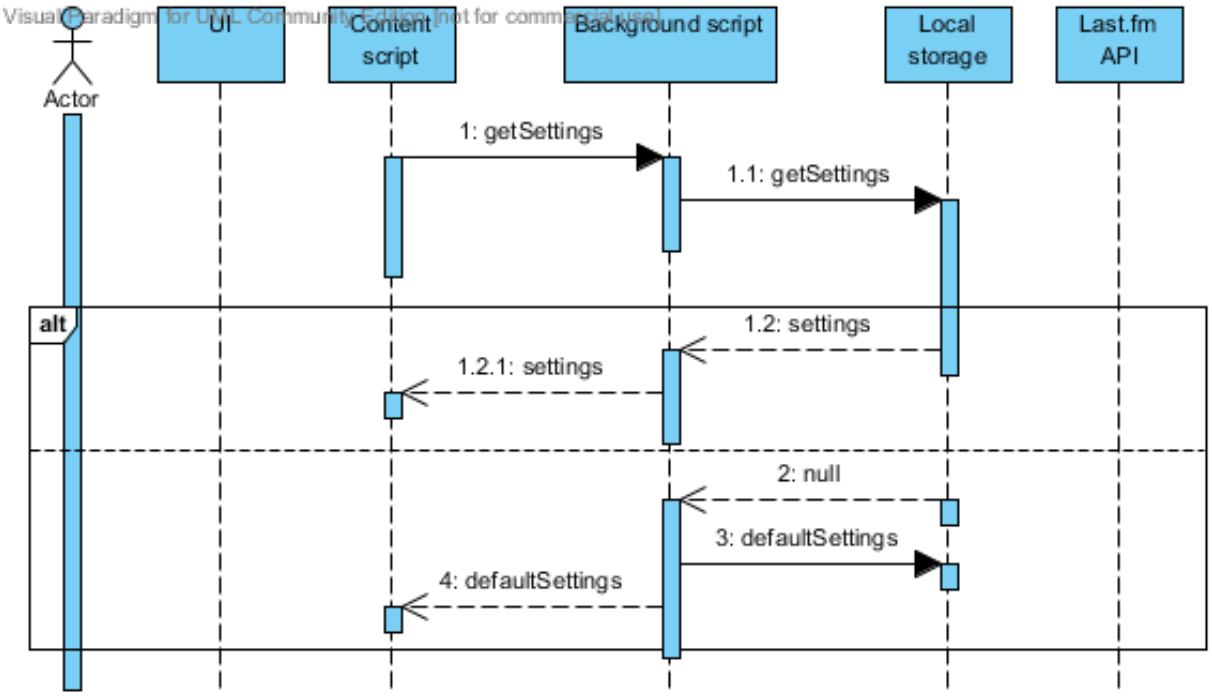


Figure 5.3: Sequence diagram: opening the Last.fm recommendations page part 2: retrieving stored settings.

of recommendations added to the number of top artists, and the number of neighbours involved The resulting time complexity is then $O((R+T)*N)$.

Data: Active user *user*
Result: Datastructure D
$D \leftarrow \emptyset$;
$N \leftarrow$ getNeighbours(*user*);
$U \leftarrow$ union(*user*, N);
$T \leftarrow$ getTopartists(*user*);
$R \leftarrow$ getRecommendations(*user*);
$A \leftarrow$ union(T, R);
foreach *artist a in A* **do**
  foreach *user u in U* **do**
    $Similar \leftarrow$ getSimilar(a);
    $Score \leftarrow$ compare(union(a, $Similar$), u);
    if $Score > threshold$ **then**
      $D.artistMAP[a] \leftarrow$ union($D.artistMAP[a]$, u);
      $D.userMAP[u] \leftarrow$ union($D.userMAP[u]$, a);
    end
  end
end

Algorithm 1: Loading the data for the visualization.

The corresponding sequence diagram is shown in figure 5.4. The two calls within the inner loop have a large impact on the performance of the algorithm. Cashing parts of the data structure is possible. However, small changes in the data may have an impact

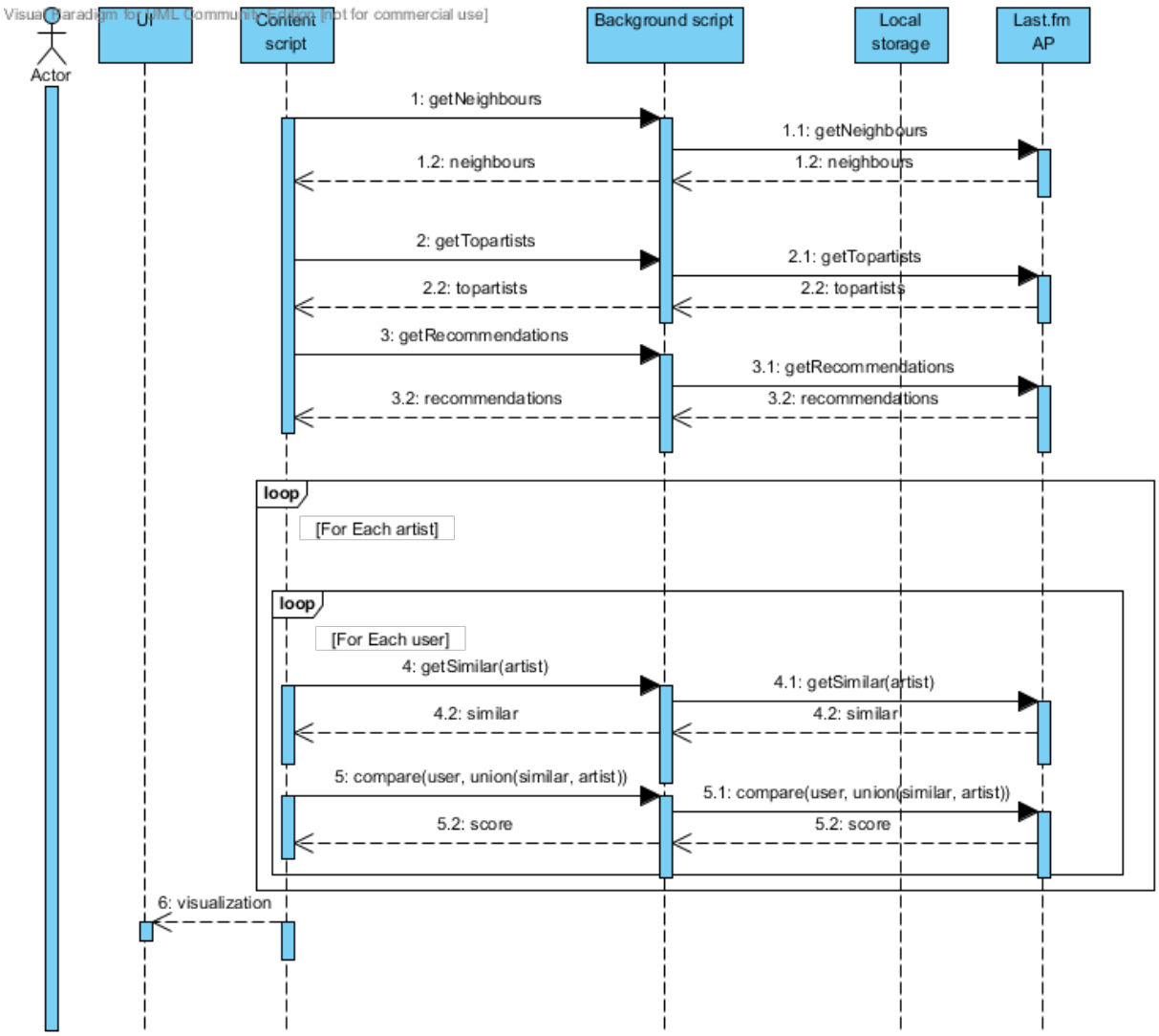


Figure 5.4: Sequence diagram: loading the visualization.

on the rest of data structure. For example if a user gets promoted to a neighbour and another gets demoted, it is impossible to know which user has to be removed from the data structure without comparing the lists of updated neighbours to the old version. Next the relevant neighbours should be removed from the data set and the new ones added. For artists that are promoted to the status of recommendation, there is a similar scenario. In this case, note that all of the users should be compared to the new items as well.

5.3 Implementation

5.3.1 Configuration file `manifest.JSON`

Aside from the basic parameters in the `manifest.JSON` file, such as `name`, `version`, `manifest_version`, et cetera, there are several parameters that require some more attention. First of all, this extension is defined as a so-called page action. This is done by

adding the page action with required attributes, namely certain icons and a default title, to the JSON file. The icon will become visible in the address bar when visiting a page defined in the `content_scripts` parameter. In this case the script will become active when visiting the recommendations page of the *Last.fm* website. The various CSS and JavaScript used in the extension are also listed together with the content script.

As the application makes use of the storage functionality, the storage option should be added to the permissions. Since the application does calls to the Last.fm audioscrobbler, this link should be added to the permissions as well. Note that SSL is required when making external calls, otherwise the application won't even be accepted when uploading it to the chrome web store. The link should also be addded to the `content_security_policy` parameter of the manifest file.

To be able to access and load images, for example in the CSS definitions, it is necessary to add these to the `web_accessible_resources` parameter.

5.3.2 The visualization `infovis`

The visualization consists out of four main parts:

- JSON data;
- JavaScript script;
- CSS style sheet;
- Custom implementation of certain methods.

Data structure

The JSON file structure is shown in listing 5.1. It consists out of a list of artists and users that were retrieved using algorithm 1. It can be generated by using the output of algorithm 1 as input for algorithm 2.

```
1  {
2      "items" :
3      [
4          {
5              "name"  : "item.SOME_ARTIST",
6              "edges" :
7              [
8                  "item.SOME_ARTIST.user.SOME_USER",
9                  ...
10             ],
11             "owners" :
12             [
13                 "SOME_USER",
14                 ...
15             ],
16             "recommendation" : BOOLEAN
17         },
18         ...
19     ],
20     "users" :
21     [
```

```
22          {
23              "name"      : "SOME_USER",
24              "active"    : BOOLEAN
25          },
26          ...
27      ]
28  }
```

Listing 5.1: The structure of the JSON file that is the input for the visualization script.

Data: Data structure D from algorithm 1, active user *active*.
Result: JSON file $JSON$ as in listing 5.1.
$JSON \leftarrow \{\}$;
foreach *Artist a in D.artistMap.keys* **do**
  artist $\leftarrow \{\}$;
  artist.put ("name", "item." + *a.name*);
  isrecommendation $\leftarrow$ false ;
  foreach *User u in D.artistMap[a]* **do**
    if *u.equals (active)* **then**
      isrecommendation = true;
    end
    foreach *Artist d in D.userMap[u]* **do**
      artist.append ("edges", "item." + *d.name* + ".user." + *u.name*);
    end
  end
  artist.put ("recommendation", *isrecommendation*);
  JSON.append (*artist*);
end
foreach *User u in D.userMap.keys* **do**
  user $\leftarrow \{\}$;
  user.put ("name", *u.name*);
  user.put ("active", *u*.equals (*active*));
  foreach *Artist d in D.userMap[u]* **do**
    user.append ("owned", "item." + *d.name*)
  end
  JSON.append (*user*);
end

Algorithm 2: Loading the data for the visualization.

The visualization script

Once the data structure has been constructed, it is plugged into the script. Listing 5.2 shows how this is done in JavaScript, assuming that the variables $LAYOUT$ and $DATA$ are known. The script uses the data to generate nodes, edges in an `SVG` element, and a list of users as `LI` elements in an `UL` element next to the visualization.

For this visualization, a D3.js hierarchical edge-bundling example by Michael Bostock was adapted. The major changes to the original code are the extension of the original data structure as explained in the previous section, and the addition of extra CSS classes to support interactions with the user list, which are discussed in the next section. In

Interaction	SVG Node in **#chart svg**	User LI in **ul#users**
Click node	*.link-item-clicked* *.node-item-clicked*	*.user-item-clicked*
Click user	*.user-clicked* *.link-user-clicked* *.node-user-clicked*	*.user-clicked*
Hover node	*.node-item-mouseover* *.user-item-mouseover*	*.user-item-mouseover*
Hover user	*.node-user-mouseover* *.link-user-mouseover*	*.user.user-mouseover*

Table 5.1: Overview of the classes that added for each supported interaction for each interaction target.

	Blue	**Green**	**Red**
Active user	*.blue-active*	*.green-active*	*.red-active*
Mouseover	*.blue-mouseover*	*.green-mouseover*	*.red-mouseover*
Click	*.blue-clicked*	*.green-clicked*	*.red-clicked*

Table 5.2: Overview of the classes that added for each supported colour.

conclusion, for a detailed description of the visualization code, we refer to the D3.js website[8].

```
1 var WHITEBOX = new Whitebox();
2 WHITEBOX.setLayout(LAYOUT);
3 WHITEBOX.setData(DATA);
4 WHITEBOX.create();
```

Listing 5.2: Create a new Whitebox object for given settings and data.

Style sheet

To support hover and click interactions, each node and each user `LI` element has an *onmouseover*, *onmouseout*, and *click* event listener attached to it. When one of these events is triggered, the appropriate classes are added or removed from these elements. Table 5.1 shows which classes are activated for which interaction. Each of these classes in also combined with another set of classes as listed in table 5.2. By changing the colour classes for nodes, edges and `LI`'s, colour patterns chosen by the end user are applied on the fly.

[8]The original code of the hierarchical edge-bundling example can be found at `http://bl.ocks.org/mbostock/1044242`.

Chapter 6

Conclusion and future work

Bibliography

[1] C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More.* Hyperion, 2006.

[2] S. Bostandjiev, J. O'Donovan, and T. Höllerer. Tasteweights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 35–42, New York, NY, USA, 2012. ACM.

[3] M. Bostock. D3.js - data-driven documents. URL: `http://d3js.org/`, 2012. [Online; accessed 26-December-2012].

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

[5] T. Crnovrsanin, I. Liao, Y. Wuy, and K.-L. Ma. Visual recommendations for network navigation. In *Proceedings of the 13th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'11, pages 1081–1090, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[6] K. Dekimpe and B. Demoen. Fundamenten voor de informatica. URL: `http://people.cs.kuleuven.be/~bart.demoen/FVI/fundamenten.pdf`, 2007. [Online; accessed 9-February-2013].

[7] E. Duval. Chi: evaluation. URL: `http://www.slideshare.net/erik.duval/chi-evaluation-11570071`, 2012. [Online; accessed 20-February-2013].

[8] S. Faridani, E. Bitton, K. Ryokai, and K. Goldberg. Opinion space: A scalable tool for browsing online comments. *CHI 2010 : Understanding Comments*, 2010.

[9] Google. Zoeken met afbeeldingen - inside search - google. URL: `http://www.google.com/insidesearch/features/images/searchbyimage.html`, 2011. [Online; accessed 8-February-2013].

[10] Google. Background pages - google chrome. URL: `http://developer.chrome.com/extensions/background_pages.html`, 2012. [Online; accessed 28-December-2012].

[11] Google. chrome.browseraction - google chrome. URL: `http://developer.chrome.com/stable/extensions/browserAction.html`, 2012. [Online; accessed 28-December-2012].

[12] Google. Formats: Manifest files - google chrome. URL: `http://developer.chrome.com/stable/extensions/manifest.html`, 2012. [Online; accessed 28-December-2012].

[13] Google. Google chrome extensions. URL: `http://developer.chrome.com/extensions/index.html`, 2012. [Online; accessed 28-December-2012].

[14] Google. Overview - google chrome. URL: `http://developer.chrome.com/extensions/overview.html`, 2012. [Online; accessed 28-December-2012].

[15] L. Gou, F. You, J. Guo, L. Wu, and X. L. Zhang. Sfviz: interest-based friends exploration and recommendation in social networks. In *Proceedings of the 2011 Visual Information Communication - International Symposium*, VINCI '11, pages 15:1–15:10, New York, NY, USA, 2011. ACM.

[16] B. Gretarsson, S. Bost, C. Hall, and T. Höllerer. Smallworlds: Visualizing social recommendations. *Eurographics/ IEEE-VGTC Symposium on Visualization 2010*, 2010.

[17] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[18] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[19] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, Jan. 2000.

[20] O. C. Herrada. Music recommendation and discovery in the long tail. URL: `http://mtg.upf.edu/static/media/PhD_ocelma.pdf`, 2008. [Online; accessed 26-April-2013].

[21] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, Sept. 2006.

[22] D. Holten and J. J. V. Wijk. Force-directed edge bundling for graph visualization, 2009.

[23] T. jQuery Foundation. jquery. URL: `http://jquery.com`, 2013. [Online; accessed 13-May-2013].

[24] T. jQuery Foundation. jquery. URL: `http://jqueryui.com`, 2013. [Online; accessed 13-May-2013].

[25] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, Jan. 2002.

[26] J. Kirakowski. Questionnaires in usability engineering. URL: `http://www.ucc.ie/hfrg/resources/qfaq1.html`, 2000. [Online; accessed 20-February-2013].

[27] G. Klein, B. Moon, and R. R. Hoffman. Making sense of sensemaking 1: Alternative perspectives. *IEEE Intelligent Systems*, 21(4):70–73, July 2006.

[28] G. Klein, B. Moon, and R. R. Hoffman. Making sense of sensemaking 2: A macrocognitive model. *IEEE Intelligent Systems*, 21(5):88–92, Sept. 2006.

[29] V. Krebs. "2012 political book network". URL: `http://www.thenetworkthinkers.com/2012/10/2012-political-book-network.html`, 2012. [Online; accessed 6-May-2013].

[30] KULeuven. "masterproef t313 : Visualisatie van muziekaanbevelingen". URL: `https://www.cs.kuleuven.be/cs/studenten/eindwerken/20122013/onderwerpen/individueel/T313.shtml`, 2008. [Online; accessed 10-October-2012].

[31] Last.fm. Faq - last.fm. URL: `http://www.last.fm/help/faq?category=99`, 2012. [Online; accessed 13-May-2013].

[32] Last.fm. Last.fm - listen to internet radio and the largest music catalogue online. URL: `http://www.last.fm/`, 2012. [Online; accessed 28-November-2012].

[33] M. Levy and K. Bosteels. Music recommendation and the long tail. URL: `http://womrad.org/2010/papers/1.pdf`, 2008. [Online; accessed 26-April-2013].

[34] T. Li and M. Ogihara. Toward intelligent music information retrieval. *Trans. Multi.*, 8(3):564–574, Sept. 2006.

[35] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[36] Microsoft. Svg - internet explorer 9 guide for developers. URL: `http://msdn.microsoft.com/en-us/ie/hh410107.aspx`, 2012. [Online; accessed 26-December-2012].

[37] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[38] J. Nielsen. Thinking aloud: The #1 usability tool. URL: `http://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/`, 2012. [Online; accessed 20-February-2013].

[39] J. Nielsen. Why you only need to test with 5 users. URL: `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`, 2012. [Online; accessed 20-February-2013].

[40] C. North. Toward measuring visualization insight. *IEEE Comput. Graph. Appl.*, 26(3):6–9, May 2006.

[41] J. O'Donovan, B. Smyth, B. Gretarsson, S. Bostandjiev, and T. Höllerer. Peerchooser: visual interactive recommendation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1085–1088, New York, NY, USA, 2008. ACM.

[42] M. J. Pazzani and D. Billsus. The adaptive web. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The adaptive web*, chapter Content-based recommendation systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[43] A. Rajaraman, J. Leskovec, and J. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2012.

[44] J. Sauro. Measuring usability with the system usability scale (sus). URL: `http://www.measuringusability.com/sus.php`, 2011. [Online; accessed 20-February-2013].

[45] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[46] P. Shirley and S. Marschner. *Fundamentals of Computer Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 3rd edition, 2009.

[47] C. Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces (Interactive Technologies)*. Morgan Kaufmann, 1st edition, 2003.

[48] C. Snyder. What is paper prototyping. URL: `http://www.paperprototyping.com/what.html`, 2003. [Online; accessed 10-February-2013].

[49] J. Steele and N. Iliinsky. *Beautiful Visualization: Looking at Data through the Eyes of Experts*. O'Reilly Media, Inc., 1st edition, 2010.

[50] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 801–810, Washington, DC, USA, 2007. IEEE Computer Society.

[51] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), 2002.

[52] UsabilityNet. Usabilitynet: International standards. URL: `http://www.usabilitynet.org/tools/r_international.htm#9241-11`, 2006. [Online; accessed 20-February-2013].

[53] W. W. W. C. (W3C). Implementations - svg. URL: `http://www.w3.org/Graphics/SVG/WG/wiki/Implementations`, 2010. [Online; accessed 26-December-2012].

[54] W. W. W. C. (W3C). Scalable vector graphics (svg) 1.1 (second edition). URL: `http://www.w3.org/TR/SVG/`, 2011. [Online; accessed 26-December-2012].

[55] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[56] J. S. Yi, Y.-a. Kang, J. T. Stasko, and J. A. Jacko. Understanding and characterizing insights: how do people gain insights using information visualization? In *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*, BELIV '08, pages 4:1–4:6, New York, NY, USA, 2008. ACM.

[57] S. Zhao, M. X. Zhou, Q. Yuan, X. Zhang, W. Zheng, and R. Fu. Who is talking about what: social map-based recommendation for content-centric social websites. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 143–150, New York, NY, USA, 2010. ACM.

List of Figures

List of Tables

Appendix A

Use cases

Table A.1: Use case 1 *Hover item*

Primary actor:	Active user
Preconditions:	The application has access to the active user's profile; The visualization has successfully loaded;
Basic flow:	(1) The user enters the area of an item node in the graph; (2) The system highlights the nodes and edges that are directly connected to the target node (popout technique); (3) The system highlights icons next to the graph corresponding to neighbours that have the target item in their profile; (4) The user exits the node area; (5) The system shows the default layout of the graph;

Table A.2: Use case 2 *Hover neighbour*

Primary actor:	Active user
Preconditions:	The application has access to the active user's profile; The visualization has successfully loaded;
Basic flow:	(1) The user enters the area of a neighbour icon next to the graph; (2) The system highlights the neighbour's icon; (3) The system highlights the nodes and edges that connect items that are in the profile of the selected user; (4) The user exits the icon area; (5) The system shows the default layout of the graph;

Table A.3: Use case 3 *Click item*

Primary actor:	Active user
Preconditions:	The application has access to the active user's profile; The visualization has successfully loaded;
Basic flow:	(1) The user clicks an item node in the graph; (2) The system highlights the nodes and edges that are directly connected to the target node; (3) The system highlights icons next to the graph corresponding to neighbours that have the target item in their profile; (5) The system displays additional information about the item and options in an area next to the visualization; information includes a brief introductory text and top tracks; options include the possibility to add the item to the active user's profile.
Alternate flow:	(2.a) the item was already selected: the item is now deselected;

Table A.4: Use case 4 *Click neighbour*

Primary actor:	Active user
Preconditions:	The application has access to the active user's profile; The visualization has successfully loaded;
Basic flow:	(1) The user clicks an item node in the graph; (2) The system highlights the nodes and edges that are directly connected to the target node; (3) The system highlights icons next to the graph corresponding to neighbours that have the target item in their profile; (4) The system displays additional information about the item and options in an area next to the visualization; information includes a brief introductory text and top tracks; options include the possibility to add the item to the active user's profile.
Alternate flow:	(2.a) the neighbour was already selected: the neighbour is now deselected;

Appendix B

SUS questionnaire questions

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

Appendix C

Task lists for the user tests

C.1 Task list 1: testing insight and usability

The user is given some context, i.e., the user knows he/she is using a recommender system to find new music and he/she has a number artists in his/her artist library.

1. Answer the following questions without interacting with the visualization:
   (a) Describe what you see. Which visual elements stand out? Which general structures can be identified?
   (b) What do you think the visualization does?
   (c) Which the elements of the user interface, do you think allow interaction?
   (d) What do you think will happen when you:
      - hover over an node of the graph?
      - hover over one of the users?
      - click on an item?
      - click on a user?
2. Try to interact with the visualization. Answer the following questions:
   (a) Which of the artists displayed in the graph are artist suggestions?
   (b) What are the links or edges in the visualization?
   (c) Suppose you want to add an item to your profile, what steps would you undertake?
3. Add an item to your profile. Answer the following questions:
   (a) Why did you choose that particular item?
   (b) Can you give any other reasons why you should pick this item?
   (c) Can you give reasons for choosing one of the other items?

C.2 Task list 2: testing the first version of the settings menu

1. Change the number of shown recommendations up to 20.
2. Change the threshold to 0.3.
3. Change the colours to a encoding that you like.

C.3 Task list 3: testing the performance of the evaluation system

1. Find three neighbours that are closely related to you, based on the visualization.
2. Find three recommended artists you think are interesting.
3. Explain the recommendation rationale (transparency).
4. Find a suggestion for an artist you didn't know about.
    - Would you like to check our this artist's profile and listen one or more songs by this artist (persuasion)?
    - Do you think the recommender system has made a good suggestion? Would you add it your profile (effectiveness)?
    - How does it affect your trust in the recommender system (trust)?

Appendix D

Quantified self

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
@kuleuven.be
www.kuleuven.be