KU LEUVEN

FACULTY OF SCIENCE

Visualization of music suggestions

A visual explanation system for collaborative filtering

Joris SCHELFAUT

Supervisor: Prof. dr. ir. E. Duval
Affiliation *KU Leuven Department of Computer Science*

Co-supervisor: *Dr. J. Klerkx*
Affiliation *KU Leuven Department of Computer Science*

Co-supervisor: *Prof. dr. K. Verbert*
Affiliation *Technische Universiteit Eindhoven Department of Information Systems WSK&I*

Mentor: *Dr. J. Klerkx*
Co-mentor: *Prof. dr. K. Verbert*

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

Academic year 2012-2013

KU LEUVEN

FACULTEIT WETENSCHAPPEN

Visualisatie van muzieksuggesties

A visueel uitlegsysteem voor collaboratieve filtering

Joris SCHELFAUT

Promotor: Prof. dr. ir. E. Duval
Affiliatie *KU Leuven Department of Computer Science*

Co-promotor: *Dr. J. Klerkx*
Affiliatie *KU Leuven Department of Computer Science*

Co-promotor: *Prof. dr. K. Verbert*
Affiliatie *Technische Universiteit Eindhoven Department of Information Systems WSK&I*

Begeleider: *Dr. J. Klerkx*
Begeleider: *Prof. dr. K. Verbert*

Proefschrift ingediend
tot het behalen van
de graad van Master of Science
in de Toegepaste Informatica

Academic year 2012-2013

Acknowledgements

I would like to express my appreciation of the people who have helped and supported me throughout the course of the development of this thesis.

I would like to thank my supervisors Prof. dr. ir. E. Duval, Prof. dr. K. Verbert and Dr. J. Klerkx for their assistance and guidance. I am grateful to my mentors Prof. dr. K. Verbert and Dr. J. Klerkx for reading my work and providing helpful suggestions to improve it.

I acknowledge the efforts by my assessors Prof. dr. B. Berendt and Dr. ir. Frans Van Assche to read and evaluate my work.

I would also like to thank my parents, my sister Saskia and her friend Maarten, as well as some of my friends and collegues, Sander, Nik, Prince, Wouter, Macin, Tim, Tonderai, and Carl for their continuous support.

Summary

Finding new and interesting music in the abundant supply, is a difficult and complex problem. *Recommender systems* address this by filtering out candidate suggestions from the item space based on a model of the user's taste[49].

Although many approaches exist to produce accurate recommendations, the rationale of recommender systems is often opaque towards the end user. This may cause decreased levels of acceptance of its recommendations. Herlocker et al. [19] point out that *explanation systems* can overcome this problem by providing insight into the reasoning behind suggestions.

In this thesis we will look at a new explanation system for collaborative filtering, called *SoundSuggest*. This system aims to explain music recommendations made by *Last.fm* using a graph-based approach giving an approximation of the utility matrix. The system is evaluated through a user study based on aims described by Tintarev and Masthoff[53] and properties of usability as listed by Nielsen[37]. We will investigate the quality of insight gaining based on an evaluation method developed by Chris North[40], and its effects on trust, effectiveness and persuasion of *Last.fm* recommendations. Usability and related properties are evaluated through *usability engineering* and *system usability scale (SUS)* questionnaires.

All of the test users in the study were able to describe the high level algorithm for collaborative recommendation. The test users were also able to apply gained insights to look for interesting recommendations. The average SUS score of the final iteration was 80.5, suggesting the overall perceived usability is good.

Korte Samenvatting

Het vinden van nieuwe, interessante muziek in het immense aanbod, is een lastige en tijdrovende zaak. Suggestiesystemen baseren zich op een model van de muziekvoorkeuren van de gebruiker bij het zoeken naar muzieksuggesties.

Hoewel er vele strategieën bestaan om accurate suggesties te berekenen, is de eindgebruiker soms skeptisch tegenover de resultaten. Dit kan te wijten zijn aan het feit dat het systeem niet transparant is voor de gebruiker. Herlocker et al. [19] leggen uit hoe dit probleem kan worden opgelost door middel van een uitlegsysteem.

In deze thesis stellen we *SoundSuggest* voor, een uitlegsysteem voor de collaboratieve aanbeveler van *Last.fm*. Deze applicatie visualizeert een benadering van de onderliggende utility matrix door middel van een op grafe gebaseerde aanpak.

De evaluatie van het systeem gebeurd door middel van een gebruikersstudie, gebaseerd op de doelstellingen voor uitlegsystemen van Tintarev en Masthoff[53], en eigenschappen van gebruiksvriendelijkheid, opgelijst door Jakob Nielsen[37]. De kwaliteit van inzicht wordt geëvalueerd door middel van een methode ontwikkeld door Chris North[40]. Ook het effect van inzicht op vertrouwen, effectiviteit, en overredingskracht wordt onderzocht. Gebruiksvriendelijkheid en gerelateerde problemen worden bepaald aan de hand van *usability engineering* en *system usability scale (SUS)* enquêtes.

Alle testgebruikers in de studie waren in staat om een hoogniveaubeschrijving te geven van het algoritme van collaboratieve filtering. Zij konden ook hun verworven inzichten toepassen bij het zoeken naar interessante suggesties. De gemiddelde SUS score bedroeg 80.5 en geeft aan dat de algemene subjectieve gebruiksvriendelijkheid redelijk goed is.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music catalogues for online retail have become immense over the past decades. In 2013 the iTunes music catalogue was comprised of over 26 million tracks with users downloading over 25 billion songs[2]. Today virtually anyone can create music and upload it to a music database such as *bandcamp*[1], *iTunes*[2], or *Last.fm*[3][3, 2, 32]. Well-known artists and tracks make up a very small portion of this item space, which is known as the *Long-tail phenomenon*[33]. As a result, finding new, interesting music has become a challenging task. *Recommender systems* try alleviate this problem by filtering the item repository based on a user's music taste. Taste can be modelled by analyzing user preferences and tracking user behaviour, e.g., by analyzing a user's listening history[49].

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[8, 35, 42, 43]; each of them with their own advantages and disadvantages.

There are two commonly applied recommendation strategies[43]:

- **Content-based filtering (CBF)**: Using chosen or modelled features of items to define similarity between items in the user profile and candidate suggestions;
- **Collaborative filtering (CF)**: Using overlap of item sets of each user profile to find possible suggestions in the difference of these item sets.

Although recommender systems have proven to be successful in terms of prediction accuracy, the success of a recommender system also relies on the trust in its recommendations by the end user. If the user does not know why a particular item is recommended to him, the user may be reluctant to check it out. Herlocker et al. [19] describe this issue as the *black box problem*. To improve acceptance of recommendations, they propose to build an explanation system presenting the user with a *white box model* of the recommender system rationale.

There are different ways in which explanation systems can be designed. An ambitious approach would be to explain each step of the recommendation algorithm, but this not always possible or desired. Other examples of how additional context can be provided for explainations are indicating which tracks in a user's music library are closely related to the

[1] `https://bandcamp.com/`
[2] `http://www.apple.com/itunes/`
[3] `http://www.last.fm/`

Table 1.1: Explanation aims. Table adapted from Tintarev and Masthoff [53].

Aim	**Definition**
Transparency (Tra.)	Explain how the system works.
Scrutability (Scr.)	Allow users to tell the system is wrong.
Trust	Increase users' confidence in the system.
Effectiveness (Efk.)	Help users make good decisions.
Persuasiveness (Pers.)	Convince users to try or buy.
Efficiency (Efc.)	Help users make decisions faster.
Satisfaction (Sat.)	Increase the ease of usability or enjoyment.

given recommendations, giving the system's confidence in the accuracy of the suggestions, et cetera[19].

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[4, 17, 18, 41, 62].

1.1 Thesis objective

The initial thesis objectives as described in [30] are two-fold:

1. The conduction of a literature study on techniques for the visualization of music suggestions;

2. The design, implementation and evaluation of an interactive visualization that will allow the user to gain insight into the recommendation process as well as actively steer the process.

The literature study describes recommender systems and their rationale, different visualization techniques, how users gain insight into visualization, and a number of visual explanation systems. In this context we will compose a new white box model that can be used as an explanation system for the collaborative recommendation rationale. This initial design is tested and improved through a number of iterations, resulting in an application that satisfies a number of criteria. These criteria are based on a set of evaluation properties, as described in the next subsections.

1.1.1 Evaluation properties

The explanation system will be evaluated based on seven aims described by Tintarev and Masthoff [53] listed in table 1.1. Also learnability (Learn.) and memorability (Mem.), properties of usability as described by Nielsen[37], are evaluated.

1.1.2 Evaluation methodology

Transparency is tested by evaluating insight into the recommendation process based on North's evaluation method. We will use the think aloud protocol to obtain observational

Table 1.2: System usability scale questions.

Q1	I think that I would like to use this system frequently.
Q2	I found the system unnecessarily complex.
Q3	I thought the system was easy to use.
Q4	I think that I would need the support of a technical person to be able to use this system.
Q5	I found the various functions in this system were well integrated.
Q6	I thought there was too much inconsistency in this system.
Q7	I would imagine that most people would learn to use this system very quickly.
Q8	I found the system very cumbersome to use.
Q9	I felt very confident using the system.
Q10	I needed to learn a lot of things before I could get going with this system.

data. In particular we are looking for a user to make "domain specific inferences and hypotheses"[40].

Satisfaction, efficiency, and learnability are tested through think aloud usability testing and a summative *system usabiliy scale* (SUS) questionnaire. SUS is a *Likert scale* method consisting out of 10 questions, listed in figure 1.2, to investigate the subjective usability of an application[7]. Memorability is tested by asking test users that participated in previous iterations to explain the recommender rationale again at the beginning of the test.

Trust, persuasiveness, and effectiveness are evaluated through direct feedback from the test subjects.

1.1.3 Success criteria

We aim to build a system that is accessible to non-expert users with an average to high interest in music. To achieve this, we hope to achieve positive results in terms of overall usability, learnability, memorability and transparency. By providing transparency, we hope to alleviate problems with regard to trust and effectiveness.

1.2 The visual explanation system

The white box model developed for this thesis tries to explain the recommendation rationale for collaborative filtering. *Last.fm* applies a CF-based approach to generate artist recommendations[32, 33, 58]. The network structure that is inherent to this algorithm is visuallized as a graph. Users are eliminated from the graph, and are instead represented as implicit information in the remaining edges. To retain some contextual information, users are listed next to the graph and when one of them is selected, their corresponding edges and items are highlighted.

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[4] and

[4]A library using SVG, HTML and JavaScript[5]; available at: `http://d3js.org/`

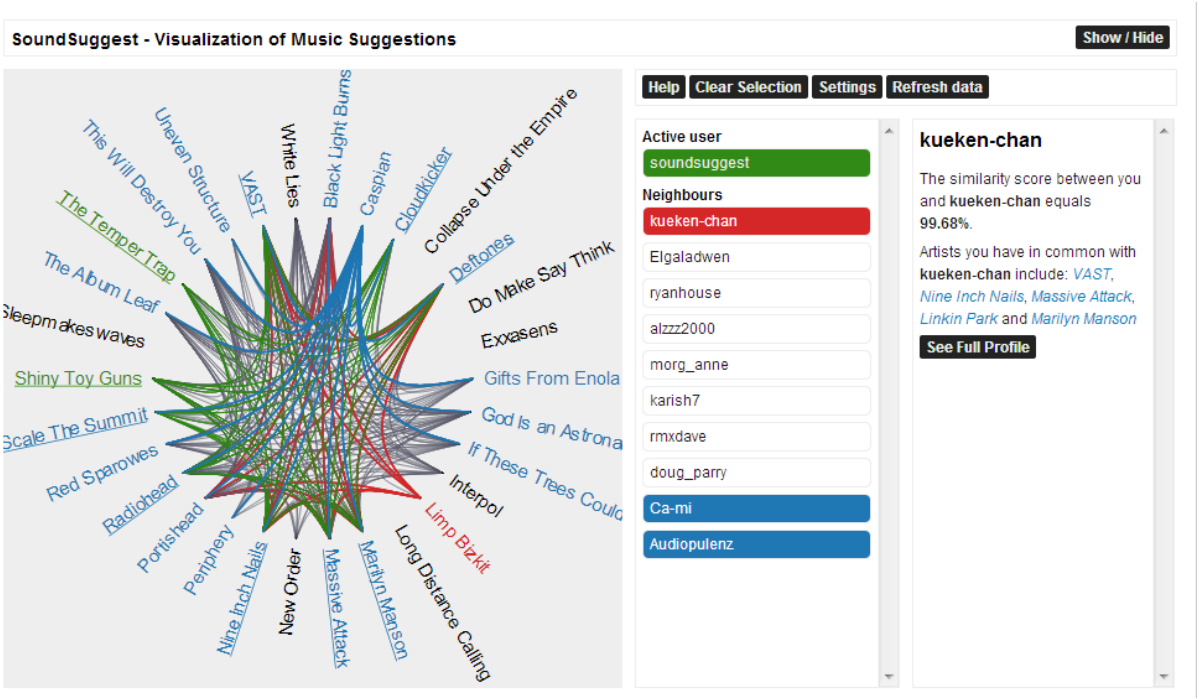


Figure 1.1: The *SoundSuggest* application.

jQuery[5], as well as a specific JavaScript library by Felix Bruns[6] to facilitate the usage of the *Last.fm API*[7].

Figure 1.1 shows the application. The application can be found in the Google Chrome web store[8]. The source code for all the projects that have been developed for this thesis is avilable at `https://www.github.com/soundsuggest/`

1.3 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 target audience and how the application can be used. Chapter 4 describes the testing methodology and the different iterations. Chapter 5 looks at the 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.

[5]Available at: `http://jquery.com/`
[6]Available at: `https://github.com/fxb/javascript-last.fm-api`
[7]Available at: `http://www.last.fm/api`
[8]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 [4, 8, 19, 35, 22].

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, 22]. 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, 22].

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, 22].

[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 [20] and [45], 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 [45].

- **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"[45]. 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[45]. 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 [8] and [22], 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)[19, 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, 8, 49].

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[8, 19]. User profiles consist of a vector of items and their ratings, which are continuously updated as the user interacts with the system over time[8].

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, e.g. *cosine distance*, *Pearson correlation*, *Tanimoto-Jaccard*[56]. For example, a small cosine distance will most likely correspond to a high similarity between profiles[43]. The discussion of the mathematics behind each of the algorithms is beyond the scope of this thesis.

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[8, 42].

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

Figure 2.2: The utility matrix A.

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*, i.e., 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 [4] 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[11].

Mathematical models for music also allow for feature extraction. Algorithms have been developed to classify music based on content features[34, 54]. 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[8]. In [8] 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, and not all of them have been implemented[8].

A good hybrid model is likely to outperform any single approach, as weaknesses of one strategy may be cancelled out by the advantages of another[49].

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

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.

The *cold start* and *gray sheep* problem both affect prediction accuracy. The *black box* problem is more closely related to perceived accuracy and trust in the recommender system.

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 [8]:

- **New User**: Finding recommendations for users with a limited rating history is hard. Since user profiles tend to build up over time, new users usually fall in this category. As a result, recommender accuracy tends to be lower for new users.
- **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[8]. 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'[8, 19]. 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[8].

Classifying users is typically a harder problem than classifying artists. An artist can usually be classified into one particular genre, whereas two users may have one genre in common, but may have rather conflicting tastes as well[43].

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[62]. This problem may decrease the acceptance by the user of item suggestions. One of the solutions for this problem, proposed by Herlocker et al. in [19], 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[19].

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

2.1.4 Music recommendation

Examples

A number of popular recommender systems for music exist, such as *Last.fm*[5], *Spotify*[6], *Grooveshark*[7], *Deezer*[8], *Tastekid*[9], *Pandora*[10] and so on.

Last.fm is an example of a music recommender system that is based on the collaborative filtering[33], although it uses additional content-based approaches as well[49]. Also *Pandora*, *Allmusic*[11] and *Shazam*[12] use hybrid approaches[49].

Bostandjiev et al. [4] have built a music recommender that uses *Wikipedia*[13], a content-based source, *Facebook*[14], a collaborative source, and *Twitter*, an *expert-based!recommender algorithm* or *context-based* source.

Profile generation

One of the difficulties for any recommendation algorithm is the modeling of good user and item profiles. Song et al. [49] list several approaches for music recommendation.

A typical *user profile* exists out of two types of data: a collection of demographic, geographic and/or psychologic data, and user listening experience[49].

Music libraries are often constantly changing, as new music is added to the user profile, listening patterns change and listening history is updated. As a result, music recommenders should be able to address this kind of *dynamic evolvement*[49]. For example, *Last.fm* deals with this problem through their *Scrobbler*, which tracks the user's listening history through various media players[31].

In [49] three types of data are associated with *item profiles*:

[5] `http://www.last.fm/`
[6] `http://www.spotify.com/`
[7] `http://www.grooveshark.com/`
[8] `http://www.deezer.com/`
[9] `http://www.tastekid.com/`
[10] `http://www.pandora.com/`
[11] `http://www.allmusic.com`
[12] `http://www.shazam.com`
[13] `http://www.wikipedia.org/`
[14] `https://www.facebook.com/`

- **Editorial metadata** (EM): for example composer, performing artist, title, genre, cover art, et cetera.
- **Cultural metadata (CM)**: data obtained from analysis of text-based sources, enabling a categorization of music; for example tag-based feature extraction as described in section 2.1.2.
- **Acoustic metadata (AM)**: data directly obtained from analysis of audio signals; for example wavelet-based approaches described section 2.1.2.

As described in section 2.1.2, item profiles can be used for content-based item recommendation.

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[46]:

- **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. Quantative data occurs in all kinds of music and user metadata.
- **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. Most editorial and cultural metadata can be considered categorical.

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, 46]. Both nodes and edges can have associated attributes. These attributes can again be either of quantitative, ordered, or categorical nature.

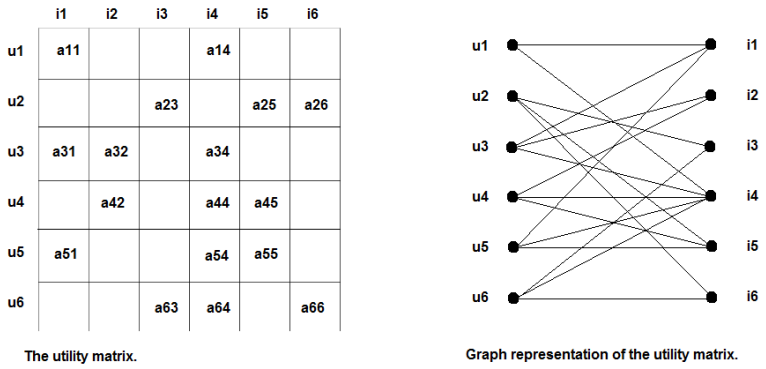


Figure 2.3: Transforming the utility matrix into a dual graph: two distinct sets of nodes, users and items, only share edges between nodes of different sets.

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 $U \cap I = \emptyset \wedge E \subseteq U \times I$[9]. Each non-blank entry in the utility matrix will then correspond to an edge. Figure 2.3 shows the dual graph of the corresponding utility matrix.

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[46].

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[46]:

- **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 [46]. Figure 2.4 gives an overview of the performance for each category, adapted from [46]. Note that spatial position is the most accurate for each data type[46].

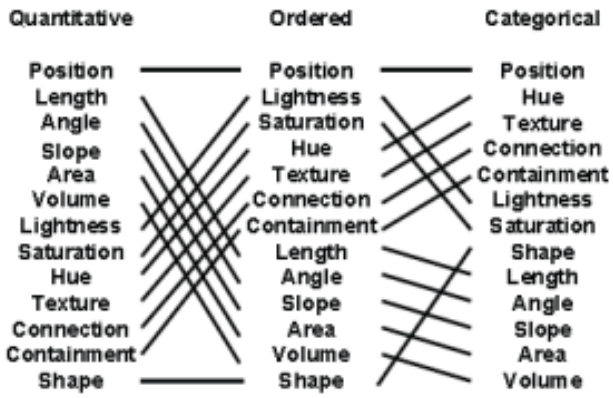


Figure 2.4: Visual encoding performance for each data type, ordered from best to worst.

Colour

In [46] 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. As a rule of thumb, a maximum of eight hue categories should be used in a visualization. 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 [46].

In the *SoundSuggest* application, items in a list of neighbours are highlighted using the hue visual channel to distinguish between three 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[46]:

- **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, elements in the vsiualization may overlap when edges cross.

- **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, 46, 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*relational data* is data that has an inherent relation among its elements[46]. The graph drawing problem describes the problem of how nodes and edges are visualized on a display[21].

Scalability is one of the central issues with graph drawing, as graph size poses several important challenges[21]. 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[21, 46]. 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 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.

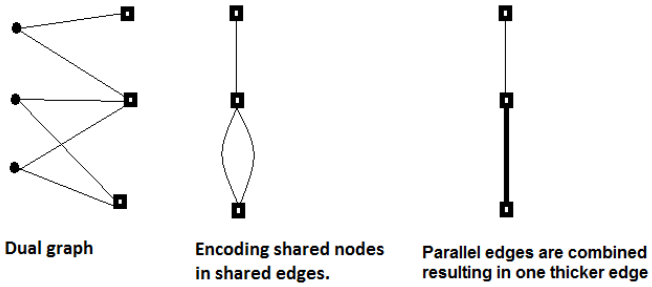


Figure 2.5: A row reduction operation on each pair of edges in a dual graph will result in a dimensionality reduction where one set of nodes is removed from the graph. An additional data reduction can be achieved by clustering edges into a thicker edge. Edge thickness then depends on the number of edges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

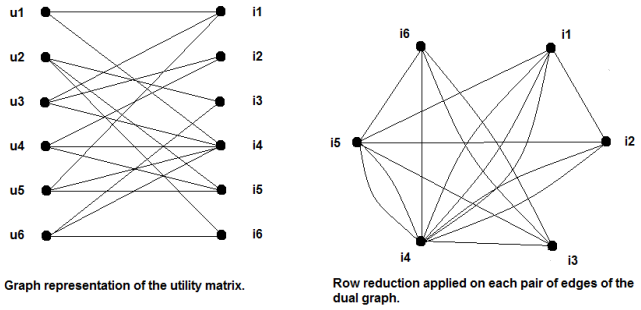
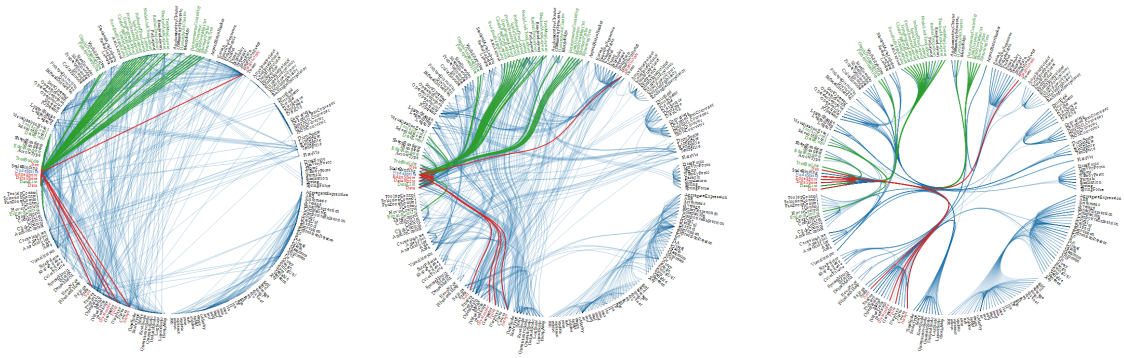


Figure 2.6: Row reduction applied on the graph in figure 2.3.



(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.

Clutter reduction

Based on a survey by Herman et al. [21], and papers by Holten [23], and Holten et al. [24], 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[23]. Edges are modelled as "flexible springs that can attract each other, similar to how electrical wires are bundled within a network"[24]. 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 can be 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[21, 23].

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

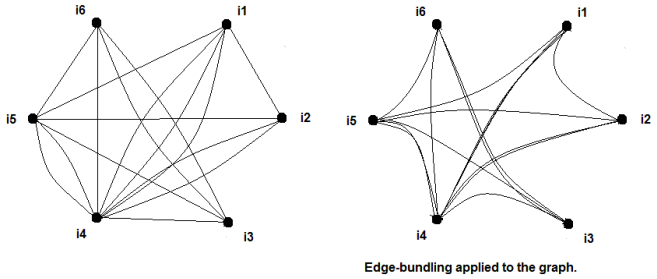


Figure 2.8: Edge bundling applied on the graph in figure 2.6.

The following subsections try to establish an answer to these questions. Most of the ideas in these subsections are drawn from papers by Yi. et al. [61], 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 [61] 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 "a critical metric for the value of importance of the reported insights in the domain"[40].

Sensemaking

Sensemaking plays an important part in insight gaining [61]. The definitions for sensemaking may vary. We adapt the definition presented by [27] and [61].

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 [61], 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[21, 57]. In order to 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 [61]. Yi et al. [61] 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[61].

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. [61] 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, careless 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[61]. 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. An important consideration is that the interval between the human action and the result on the screen should be low, promoting the so-called *priciple of transparency*. The objective of transparency is that the user will have the illusion of directly manipulating the data on the screen.

Similarly visual encodings that are counter-intuitive will also increase the cognitive load. Other barriers on insight gaining are clutter, occlusion and data overload[61]. 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[61]. 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"[55].

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 [46], 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"[46]. 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, 46, 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, 61].

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[21, 46].

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

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].

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. Ware and Mitchell state that "if the degree of relevance algorithm can reduce the visual search to around 10 to 20 items, then the gain in cognitive efficiency can easily be an order of magnitude"[57]. They point out even though that degree-of-relevance highlighting can be used for data displays with over a thousand items, to be efficient the number of members in the highlighted patterns should be much lower[57].

Table 2.2: A comparison of the visual explanation systems, based on the criteria by Tintarev and Masthoff listed in [53].

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

2.4 Visual explanation systems

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

First we will give an overview of a number of objectives for explanation systems listed by Tintarev and Masthoff in [53]. A general comparison of these systems is made, based on the previously mentioned aims. Next some additional information is given for each application.

2.4.1 Comparing visual explanation systems for item recommendation

To compare explanation systems, Tintarev and Masthoff [53] identified a set of aims for explanation systems. Table 1.1 gives an overview of these aims.

Tintarev and Masthoff explain that some of these aims are hard to reconcile. For example, a system that focuses on persuasion and efficiency, is likely to score poorly for effectiveness, as the user may be quicker to try something he/she may not like. As a result they indicate that the performance for these metrics depends on the overall system goal[53].

Table 2.2 shows which of the characteristics described by Tintarev and Masthoff, were pursued for each of the explanation systems. All of the presented systems aim to provide transparency into the recommender system through a visualization. Four out of the five systems listed here, allowed the user to make changes to the recommendation process, adding scrutability. Persuasiveness is a measure that was not explicitely covered in these studies. Most of the studies tried to find out if the recommendations found, using the visualization, were effective. Also satisfaction, trust and effciency were evaluated in some of the studies.

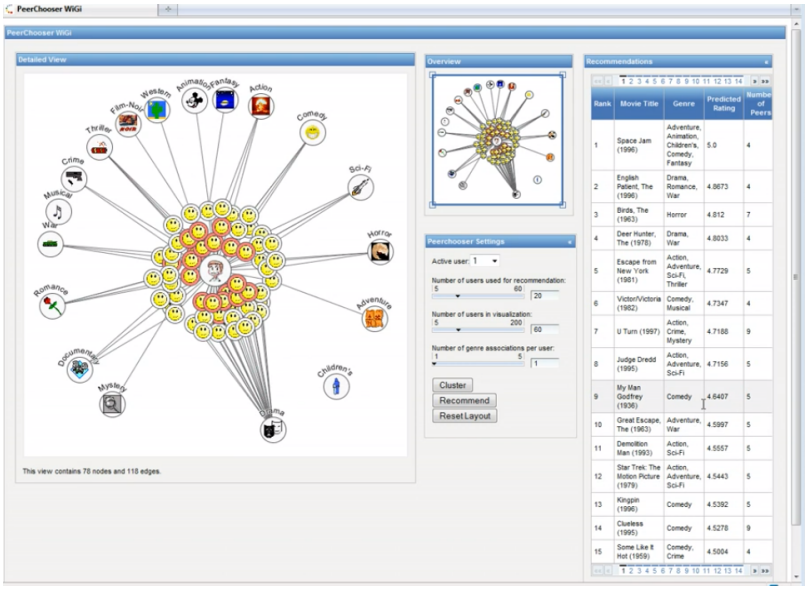


Figure 2.9: The PeerChooser interface.

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

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 visualization. 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].

2.4.3 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[62], as described earlier in section 2.1.3.

As they hope to overcome previously defined problems, Zhao et al. [62] 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.

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

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.

2.4.4 SFVis

SFVis (Social Friends Visualization) is an application developed by Gou et al. [17] 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. Figure 2.11 displays what the resulting visualization may look like.

The SFVis framework transforms a data model consisting out of social tags and social networks, into a visual form. Users can both manipulate the input and visuals on demand, adding scrutability.

The visualization is constructed as follows. 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[17].

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

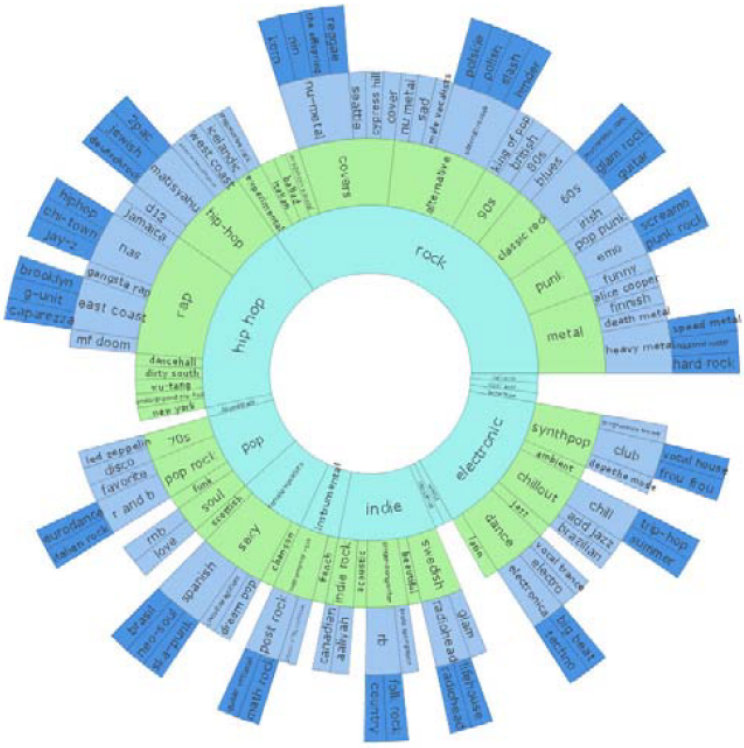


Figure 2.11: SFViz graphical user interface: tag tree.

2.4.5 Smallworlds

In [18], 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[18].

SmallWorlds is "a visual interactive graph-based interface that allows users to specify, refine and build item-preference profiles"[18]. 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[18].

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;
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.

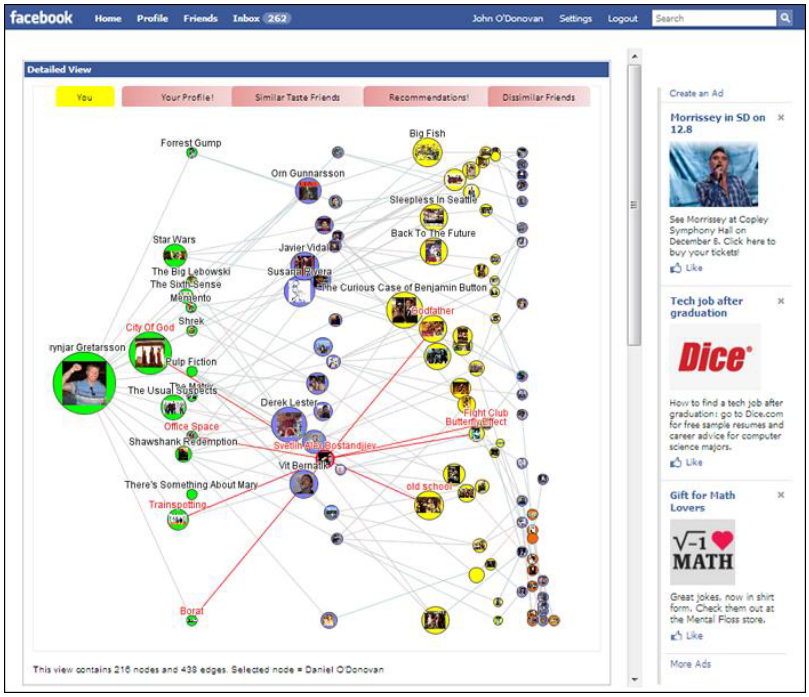


Figure 2.12: The SmallWords interface.

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.

2.4.6 TasteWeights

TasteWeights is a hybrid recommender with an interactive graphical user interface[4]. 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"[4] (cf. Smallworlds). They introduce two enhancements for the traditional techniques. Multiple web sources, namely from *Facebook*[15], *Twitter*[16] and *Wikipedia*[17] 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:

- **Profile layer**: liked items on *Facebook*;

[15] `https://www.facebook.com/`
[16] `https://twitter.com/`
[17] `http://www.wikipedia.org/`

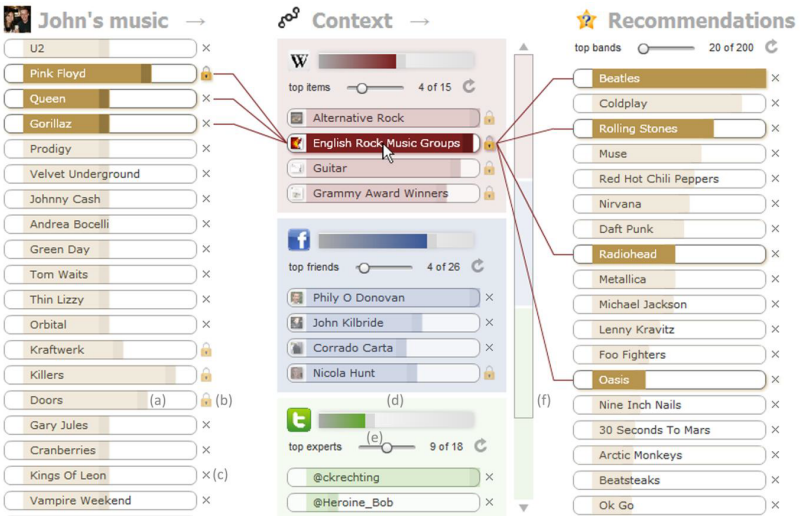


Figure 2.13: The TasteWeights interface.

- **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[4].

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.

The resulting idea for the visualization is shown in figure 2.14. This will serve as the starting point for the iterations described in chapter 4.

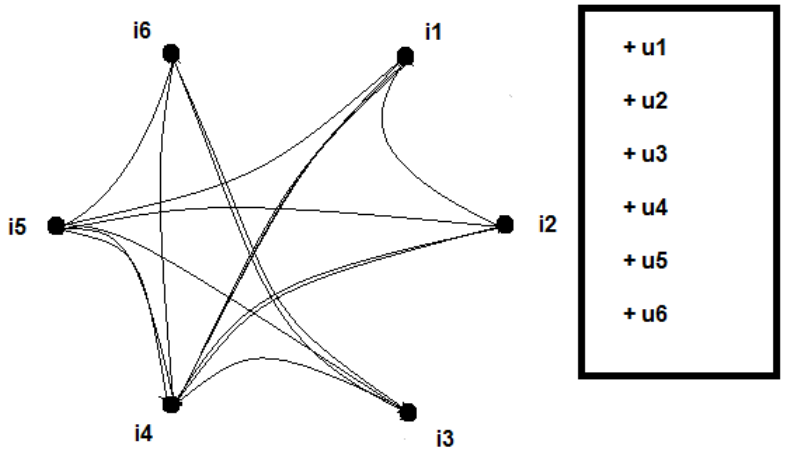


Figure 2.14: The resulting visualization serving as a white box model for collaborative filtering.

Table 2.3: The objectives for the SoundSuggest application with respect to Tintarev and Masthoff's aims. A ± indicates that this aim is not a priority, based on the success criteria listed in section 1.1.3.

Tra.	Scr.	Trust	Efk.	Pers.	Efc.	Sat.
+	−	±	±	±	±	+

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 [53]. We hope to obtain interesting results based on these criteria for the previously decribed model. The objectives for the *SoundSuggest* application with respect to Tintarev and Masthoff's aims are listed in table 2.3.

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. In [49], a classification of music listeners is given. This classification by Jennings categorizes users, aged 16-45, in one of four groups, as listed in table 3.1.

Table 3.1: Categorization of music listeners by Jennings, adapted from [49].

Type	**Percentage**	**Features**
Savants	7	Everything in life seems to be tied up with music. Their musical knowledge is extensive.
Enthousiasts	21	Music is a key part of life but is also balanced by other interests.
Casuals	32	Music plays a welcome role, but other things are far more important.
Indifferents	40	They would not loose much sleep if music ceased to exist, they are a predominant type of listeners of the whole population.

It is clear that indifferents are likely to have little interest in receiving particular artist recommendations, let alone finding out how the recommendations were computed. The focus of the application is mainly on enthousiasts and savants, as these users are more likely to look actively for music. These listeners are also more likely to look for music down the *tail*[49], cf. section 2.1.

Table 3.2 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.

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.*

Table 3.2: 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; • Has basic proficiency in 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; • Uses applications such as *iTunes*, *Windows Media Player*, and *Spotify* to listen to and purchase music; • Has used recommender systems before.
Interests:	Can be classified as a music enthousiast or savant.
Demography:	• Aged between 16 and 45 years old; • Both male and female users.

- 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 User story

The following user story tries to establish a context in which the application might prove useful. It build on the target audience, defined earlier in table 3.2.

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? For this purpose you could use a recommender system.

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? Suppose you want to find the best ones first. 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.

3.3 Story board

The story board of the application is shown in figure 3.1. It further elaborates a particular use of the application.

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.

(a) (b)

(c) (d)

Figure 3.1: A selection of the screens used in the user study with paper prototype.

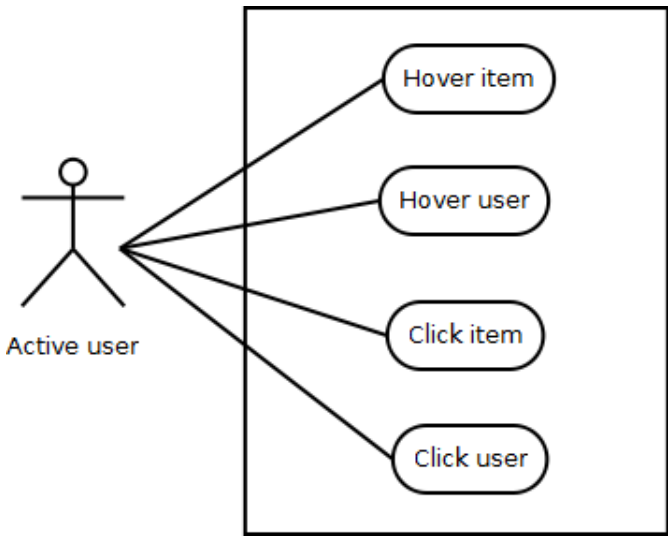


Figure 3.2: Use case diagram of the *SoundSuggest* application.

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"[48]. It is a technique for designing, testing, and refining user interfaces [47], and is closely related to usability testing [48]. In the last decade it has become a regularly applied technique in major businesses such as IBM, Digital, Honeywell, and Microsoft among others[47].

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 real data. In such prototypes, the effects of performance of algorithms and interface responsiveness can be analyzed in greater detail.

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[10].

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[10].

The explanation system will be evaluated based on seven aims described by Tintarev and Masthoff [53] listed in table 1.1. Also learnability and memorability, properties of usability as described by Nielsen[37], are also evaluated. An insight evaluation method developed by Chris North [40] is used to measure transparency, as described in section 2.3.1. Usability evaluation methods are used to measure satisfaction, efficiency, learnability and memorability. Trust, effectiveness, and persuasiveness are also evaluated during the user study. Scrutability is not supported by the explanation system.

Both usability and insight gaining methods are 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, in terms of perceived learnability and satisfaction. 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[10, 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[47].

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[10, 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.

Usability engineering

In [10], 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[10].

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'[10].

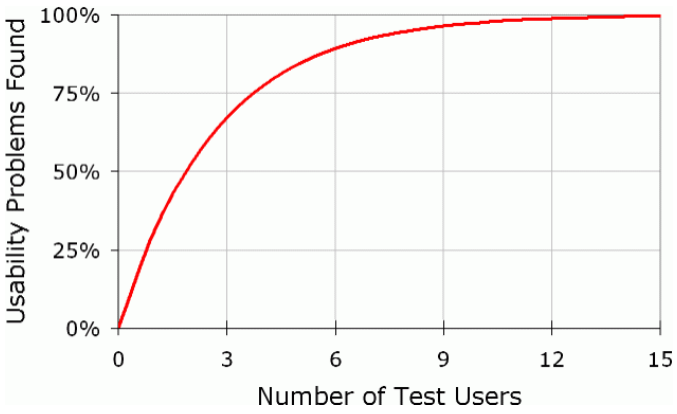


Figure 4.1: The curve shows the user test's diminishing returns beyond a certain amount of test users; adapted from [39].

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.

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 [47].

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[44]. The ten questions are

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.

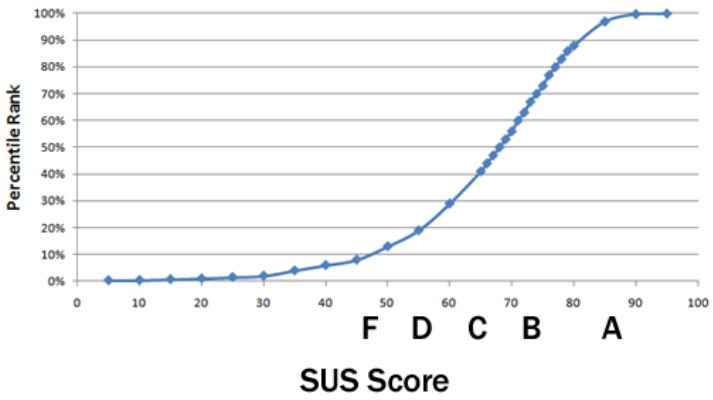


Figure 4.2: Percentile ranks associate with SUS scores and letter grades, adapted from [44].

listed in table 1.2.

To interpret SUS results, the scores should be compared on a percentile rank. Figure 4.2 shows a graph by Jeff Sauro adapted from [44]. A SUS score of 80 means then that the perceived usability is higher than almost 90 percent of the products tested in Sauro's user studies.

4.2 Iterations

We will now give an overview of the setup for the user studies. In the next subsections we will discuss each iteration in more detail.

To increase domain relevance, it is important to have representative test users. The participants were selected from the campus and among acquaintances and were between 21 and 27 years of age. For all of the iterations combined, a total of 15 users participated in the user study of whom 12 were male and 3 were female. Based on the categorization in table 3.1, 5 users identied themselves as *savants*, 7 as *enthousiasts*, and 3 as *casuals*. Although they had a notion of what a recommender system was, none of them knew how recommendation algorithms worked.

Table 4.3: The distribution of test users used in the evaluations for each iteration.

	Iteration			
	1	2	3	4
Number of users	5	5	5	10
From previous iterations	-	2	3	5

Table 4.4: The explanation aims that were evaluated in each iteration.

Aim	**Iteration**			
	1	2	3	4
Tra. Sat., Efc., Learn.	x	x	x	x
Mem.		x	x	x
Trust, Efk., Pers.				x

The test users were spread among four different iterations. Some of the users took part in multiple tests. The distribution of users can be seen in table 4.3. There are two particular incentives to let users participate in more than one test:

1. These users can provide direct feedback on changes made between the iterations that addressed certain usability issues.

2. As insight builds up over time, it would be interesting to go beyond the scope of a single user test that only lasts 30 to 60 minutes. If given the opportunity, users may develop new ways to apply their previously obtained insights, amplifying its deep, serendipitous, complex and qualitative aspects.

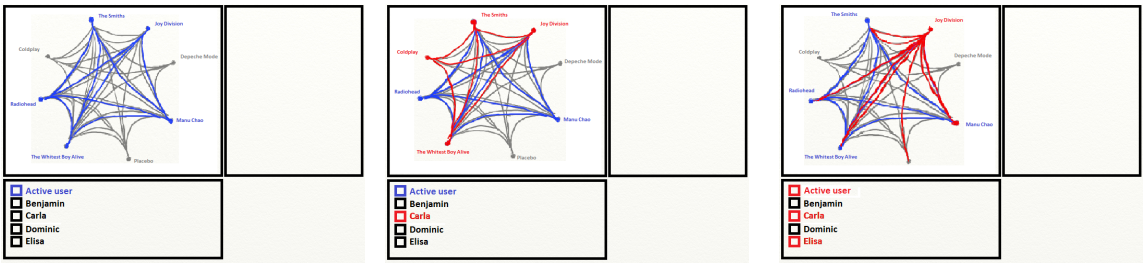
Over the four iterations the application was incrementally improved. Table 4.4 gives an overview of which aims were evaluated for the prototype in each iteration. The reason why trust, effectiveness, and persuasion are only tested in the last iteration, was that in iteration 1 and 2 no data from the active user was processed. As a result, it would be rather pointless to test for effectiveness if the user profile does not correspond to the active user's taste. In iteration three, we focus mainly on usability aspects.

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 were made of this drawing and by changing certain parts of the visualization, transitions within the visualization could be mimicked as users interacted with the screen. A selection of these screens are shown in figure 4.3.

When a user would hover an artist or user name, the screen would be replaced with the corresponding new state of the screen. To ensure a certain degree of freedom, all possible



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

(b) Clicking/hovering a user list element.

(c) Clicking/hovering an item node. When clicking a node, an additional screen would be shown next to the graph. On this screen artist information is displayed.

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

screen states were drawn for the given graph. To keep this amount manageable within the scope of a relatively short user test, the number of artists was limited to eight, and the number of users to six, including the active user. Another reason for keeping graph size small, is that small graphs tend to be easier to interpret[21]. In an environment where interactions aren't as smooth as in a real implementation, this may be of significance, keeping the principle of transparency in mind.

Test parameters

Five test users between the ages of 22 and 26 were selected. Four of them identified as enthousiasts, one of them as a casual listener. Although they did not necessarily use recommendation systems to actively search for new music, they had a notion of what these systems were.

The focus of the test is on insight, i.e., transparency, and usability, covering satisfaction, learnability and efficiency:

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 B.1. The tasks description are based on a template[1] by Carolyn Snyder[47].

Test results

The average SUS score for this iteration was 77. The distribution of results for each question is shown in figure 4.4. The next paragraphs give a more in-depth analysis of the results.

[1] A *PDF* version of this template can be found at: `http://www.paperprototyping.com/downloads/Ch6_task_template.pdf`

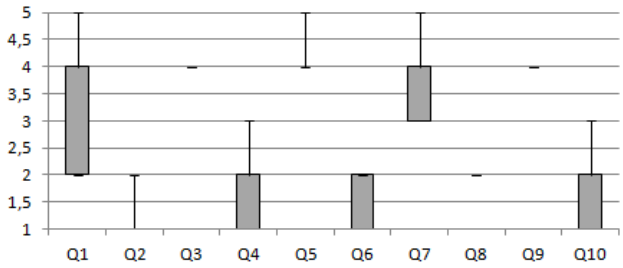


Figure 4.4: The SUS results for each question for iteration 1.

Transparency The first part of the test was aimed at forming an initial 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, illustrating insight at a deeper level. 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.

Table 4.5: Overview of the most important issues discovered in the first iteration, using the paper prototype.

Problem	Priority	Solutions
Learnability	*Medium*: May be due to limitations of the paper prototype. Also, it is expected that a minimal effort is required to learn the system, instead of instantly understanding each aspect of the application.	(1) Providing more textual clues, as suggested by test users. (2) Moving towards a digital prototype with the user's actual library as data source. As the user has a better understanding of his/her own music library, it is likely that relationships within this data can be discovered more rapidly.
Satisfaction	*Medium*: Low value of usefulness may be due to the fact that the data is not personalized. The recommendations were also not based on an actual case, but were selected in the hope to appeal to the majority of test users.	(1) Again, moving towards real, personalized data may improve this aspect of the prototype.

Satisfaction, learnability, and efficiency An average SUS score of 77 suggests that the overal usability is considered to be good - also keeping in mind some of the limitations of paper prototypes. Still, some issues remain: the results for questions `Q1`, `Q4`, `Q7`, and `Q10` were the lowest. `Q1` is corresponds to the perceived usefulness, and is related to the overall satisfaction. Scores for `Q4`, `Q7`, and `Q10` may indicate that the learnability of the application can still improve.

Once users were able to develop a strategy for choosing recommendations, they could apply it rather easily in new cases. This suggests that if the user has passed initial learning phase, the system allows for efficient decision making.

Conclusions

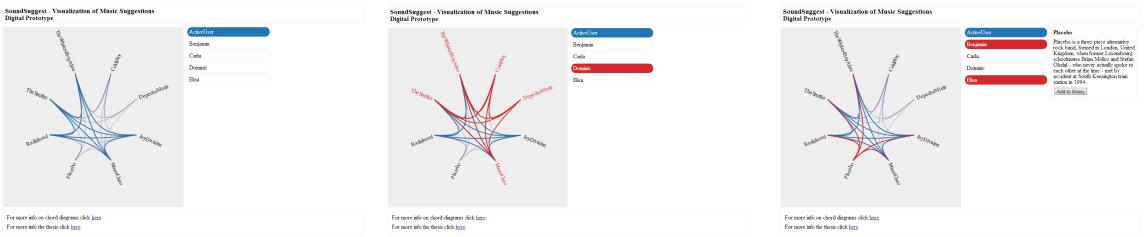
In conclusion, one user managed to get the complete mental model correct in the first step of the insight gaining process. The others were able to correct it in the second step. The model helped identifying a particular suggestion as more interesting than an others based on what they learned from the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

Table 4.5 gives an overview of the most important issues discovered in this iteration with their priority.

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 issues had arisen, we started working on the digital prototype.



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

(b) Clicking a user list element.

(c) Clicking an item node.

Figure 4.5: A selection of the screens used in the user study with the first digital prototype.

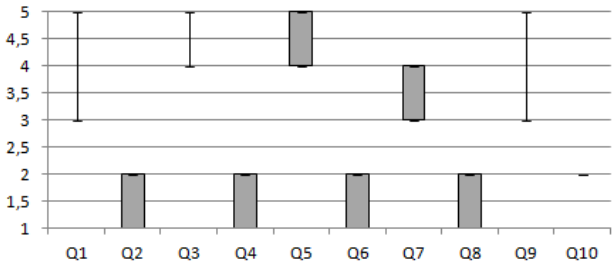


Figure 4.6: The SUS results for each question for iteration 2.

This prototype used static data, but already supported all the use cases listed in appendix A. The resulting prototype can be seen in figure 4.5.

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. Memorability was evaluated for these subjects along with the other objectives for this iteration. Based on listening habits, one savant, two enthousiasts and two casual listeners were tested.

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

The average SUS score for this iteration was 79.5. The distribution of results for each question is shown in figure 4.6. A detailed analysis of the results is given in the following paragraphs.

Transparency 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 immediately 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.

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.

Satisfaction, learnability, memorability, and efficiency Some usability issues surfaced during the think aloud test. Two users thought that deselecting an item or user should be triggered by clicking somewhere outside the graph as well. Some test users complained that the help files did not have a table of contents or some sort of overview. Two users stated that it would be nice to have some kind of overview of the meaning behind each colour. Also having an extra colour to distinguish between selected profiles or edges and hover queries would be a welcome addition.

Overall, the SUS questionnaire results stayed more or less stable. Results for question `Q7` are still rather low, indicating that users still don't see the system as very easy to learn.

In terms of efficiency the same idea as in iteration 1 applies. Of course, as interactions are much faster, this also has a positive effect on the efficiency of system usage.

Conclusions

Overall, the transition from paper to digital prototype was successful, apart from the issue with parallel edges overlapping. 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

Table 4.6: Overview of the most important issues discovered in the second iteration, using the first digital prototype.

Problem	Priority	Solutions
Parallel edges overlap	*Low*: If possible, it would be nice to see this problem solved, but it is not considered to be a priority.	(1) Developing a new graph drawing algoritm, although this is perhaps beyond the scope of this thesis. (2) Allowing to change the bundling strength might improve the overall layout of the graph.
Meaning behind colour encodings.	*High*: As there already is a significant amount of implicit information in the visualization, it may be wise to give some direct clues as well.	(1) Show a legend of the visual encodings. (2) Enable the user to choose encodings him/herself.
Deslecting an item	*Medium*: Most users did not really had a problem with this, but it seems this could be solved easily enough.	(1) Adding a button to clear the current selection. (2) Track click events outside the visualization to clear the selection.

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.

Table 4.6 gives an overview of the most important issues for iteration 2.

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. The settings menu is shown in figure 4.7.

To solve the problem of deselecting an item, an additional button titled *Clear selection* was added in the menu bar. As the data loads, a spinner indicates to the user that the system is busy, as can be seen in figure 4.8. The resulting prototype is shown in figure 4.9.

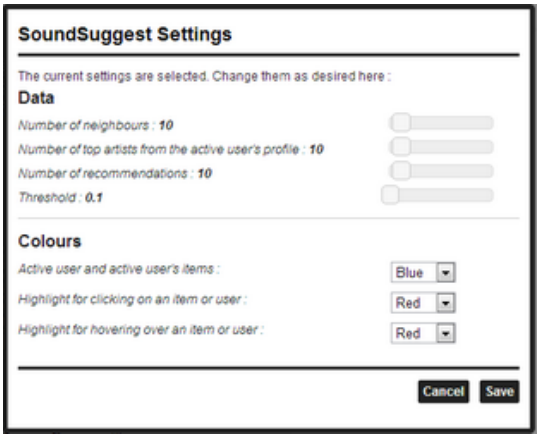


Figure 4.7: The settings menu of the second digital prototype.

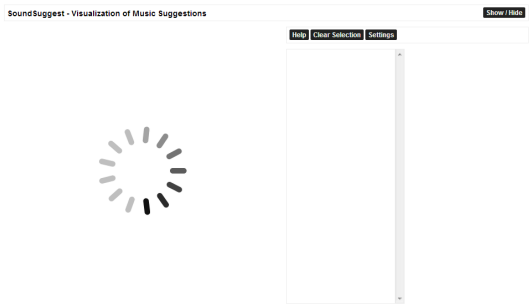
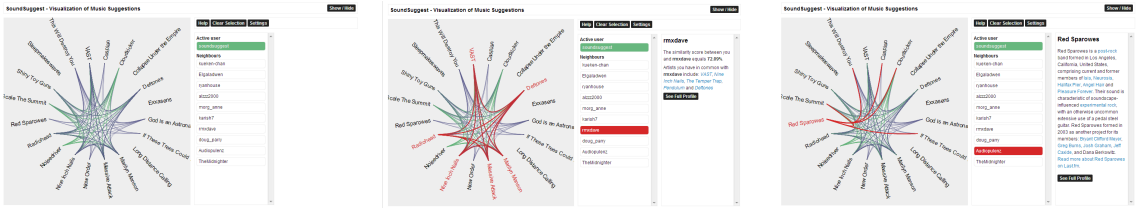


Figure 4.8: When the data is loading, a spinner is shown to indicate something is happening.



(a) The default visualization with the active user's items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Figure 4.9: A selection of the screens used in the user study with the third digital prototype.

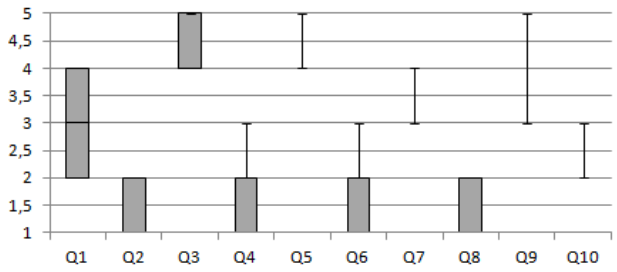


Figure 4.10: The SUS results for each question for iteration 3.

Test parameters

Five test subjects were selected, who were *Last.fm* users between the ages of 22 and 26. Three of them used the website or its *Scrobbler* on a weekly basis or even more frequently, the other two only used *Last.fm* on a monthly basis. 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 addition 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 B.2 were used to investigate these areas. Insight by new users could again be tested using the scheme in appendix B.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

The average SUS score for this iteration was 76.5. The distribution of results for each question is shown in figure 4.10.

Transparency and memorability 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. This suggests that the memorability of the explanation system is adequate. Based on previous experience and insight, one of these users explained he had found a new reasoning to select an artist based on the graph.

When testing insight for new users, no notable differences from previous iterations occurred. Theey also needed interaction with the visualization before the recommender rationale could be discovered.

Usability, satisfaction, efficiency, and learnability 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 was 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. This may also explain the fact that the average SUS score is lower than in previous iterations.

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. This also has a negative effect on the satisfaction of system usage, which may also provide an additional explanation for the lower overal score.

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.

In conclusion, the main issues from the previous iteration have been solved. Table 4.7 gives an overview of the most important issues for iteration 3.

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

The prototype

To reduce issues with distinguishability, an additional option was added to the settings menu to alter the *tension* of the edges, i.e., the edge-bundling strength. 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.

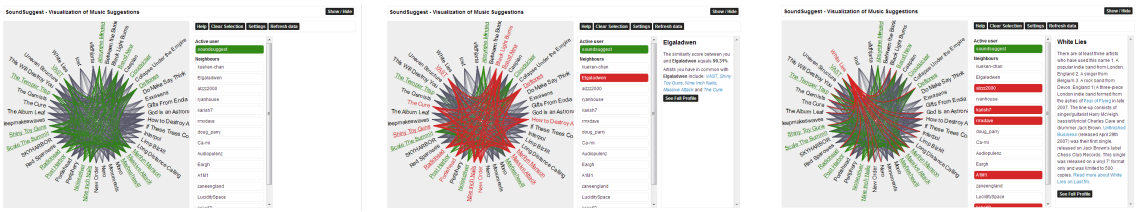
A button to refresh the data and update the visualization was added to solve the problem of having to refresh the whole page to have the latest version of the data.

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 4.11.

Table 4.7: Overview of the most important issues discovered in the third iteration, using the second digital prototype.

Problem	Priority	Solutions
Threshold	*High*: It is confusing and clearly has a negative impact on the overal user experience.	(1) Remove this functionality.
Data load speed	*High*: The main problem lies with the fact that changing the settings, reloading the page, and so on, results in reloading all the data, even though all the required data is known.	(1) Simply keeping a local copy of the latest version of the data. This version can then be updated either manually, or by checking for updates over a predefined time interval.
Visual clutter	*High*: Becomes an important problem as the graph size increases. May hinder the insight gaining process. May prohibit users from efficiently comparing user profiles and recommendations.	(1) Changing the graph layout: clustering nodes, changing edge-bundling strength, allowing to move nodes. (2) Change edge opacity to avoid visual clutter within dense regions of the graph. (3) Selectively hide elements temporary.



(a) The default visualization with the active user's items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Figure 4.11: A selection of the screens used in the user study with the third digital prototype.

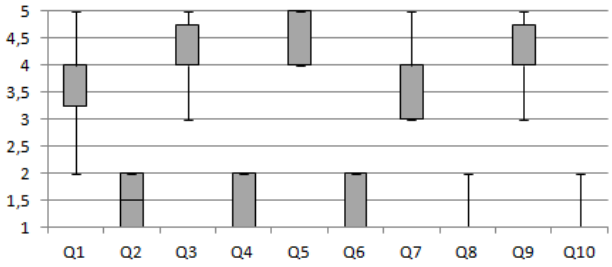


Figure 4.12: The SUS results for each question for iteration 4.

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 B.1. The tasks listed in appendix B.3 were used to test the following explanation system properties: *transparency*, *effectiveness*, *persuasion*, *trust*, and *satisfaction*.

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

The average SUS score for this iteration was 80.5. The distribution of results for each question is shown in figure 4.12.

Transparency 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, i.e., top artists in the neighbouring profiles, and the total number neighbours that own these items.*

Trust, effectiveness, and persuasion Most users already knew more or less who the recommended 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 discovering 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.

Although the explanation system was not always as effective in helping to find good recommendations, it provided an additional means for the user to establish his/her own approach for finding recommendations. For example a user would look at neighbours for artist suggestions, rather than just the artist recommendations by *Last.fm*.

Satisfaction, memorability, learnability, and efficiency 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. Although the tension parameter was visually pleasing, this did not completely resolve these scalability issues. Most users liked a tension in the area of 0.45 to 0.60 for graphs including up to 40 nodes. Beyond this amount of nodes its effects became less important.

By underlining owned artists, it had become easier to compare user profiles. Users from previous iterations stated that this was definitively an improvement, but that there might still be better ways to visualize this.

The distribution of scores for each separate SUS question for the last iteration is shown in figure 4.12. The questions with the lowest scores were the first question and the seventh question. It should be noted thaht all test users were frequent *Last.fm* users and also not all of the test users used the Chrome browser that was required for running the Chrome extension. Nonetheless there were test users that were very positive about the application in that respect as well. Although there were no negative votes for question 7, a lot of users were not convinced that using the application was easy to learn. There might still be some work left to make the application more accessible to casual users.

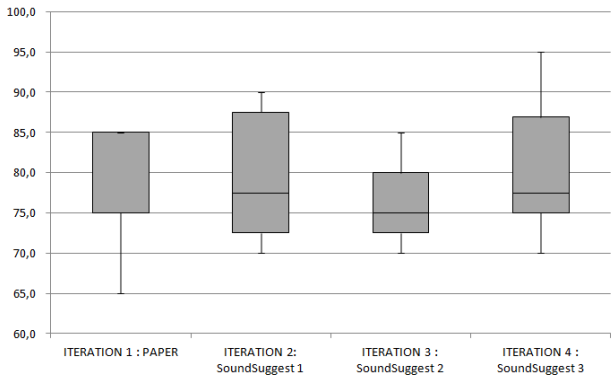


Figure 4.13: The evolution of the SUS values over the four iterations, visualized as box plots.

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.

Based from the choice of the tension parameter by users, a better default value for the tension setting would probably be around 0.55.

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

Figure 4.13 shows the evolution of the SUS values over the four iterations. Based on the percentile rank, the average score of 80.5 for the final iteration procudes a grade of `A`.

Table 4.8 gives an overview of the most important issues for iteration 4.

Table 4.8: Overview of the most important issues discovered in the fourth iteration, using the third digital prototype.

Problem	**Priority**	**Solutions**
Clutter reduction in dense graphs	*High*: Visual clutter may hinder the insight gaining process, and may prohibit the user of comparing recommendations and user profiles effectively.	(1) Sorting the artists by recommendation or top artist would make it even easier to distinguish between owned and recommended artists. (2) Clutter reduction through change in opacity between elements that are relevant and irrelevant to the selection. (3) Selectively hiding users and their corresponding edges in the graph.
Data load efficiency	*Medium*: Some initial load of data will always be necessary, so for evaluation purposes this is probably a lesser concern. To target a larger audience, this will become increasinly important.	(1) Using another algorithm that focuses less on node connectivity, but more on the quantity of data. When more items are involved, connectivity within the graph is likely to be higher as well. (2) Updating the graph as more data loads, depending on the predefined settings. (3) When refreshing the data, only update items that are new.
Learnability	*Medium*: Although the functionality of the application seems to be easy enough to learn, e.g. using the options in the settings menu, learning how the application can be useful, still takes up some time. This is probably reflected by lower scores for questions `Q1`, `Q7`, and `Q3`.	(1) Further refining labels and other visual clues for the end user.

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 the functionality of the Google Chrome web browser[15].

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[13]. 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[16].

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[14].

Many extensions use a two-layered structure in which you have a background page and UI pages or content scripts[16]. 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[12]. 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[16].

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[16].

[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[16].

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[60]. It is supported by most of the latest versions of most popular browsers, including *Chrome*, *Firefox*, *Internet Explorer 9*, *Opera* and *Safari*[36, 59].

D3.js is a JavaScript library that uses the W3C standards *HTML*, *SVG* and *CSS* to build data-driven documents[5]. 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"[51].

- **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"[52].
- **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 A, i.e., the number of

[5]

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

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

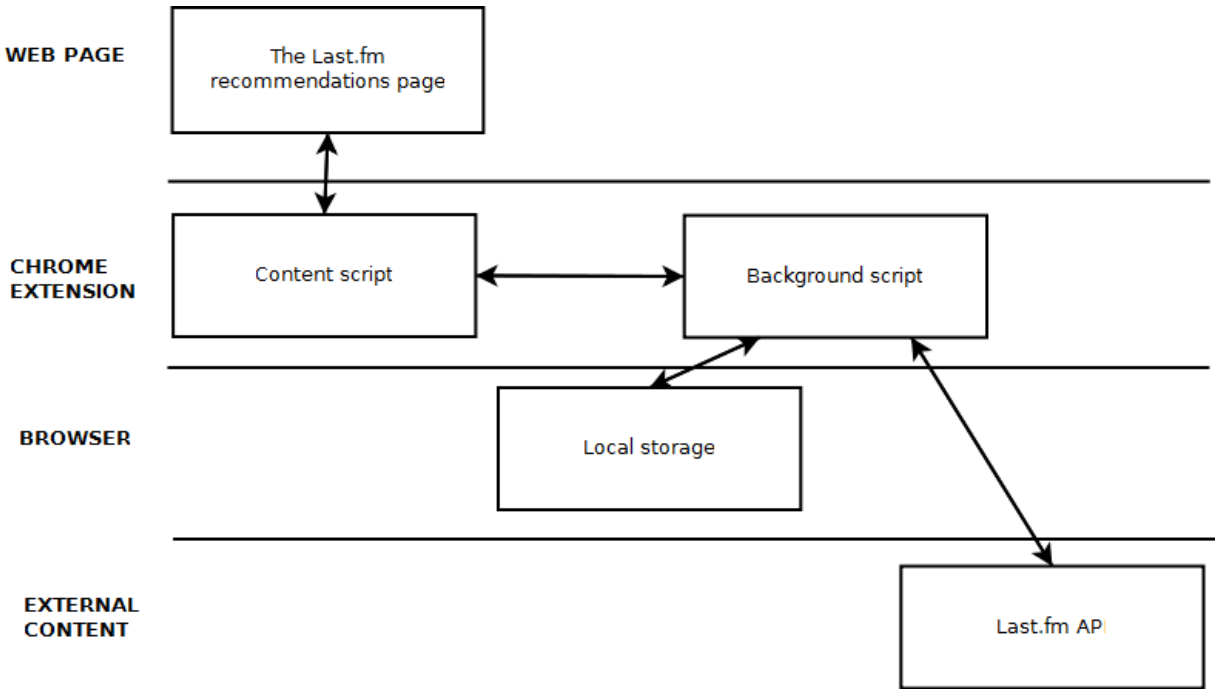


Figure 5.1: The architecture of the application.

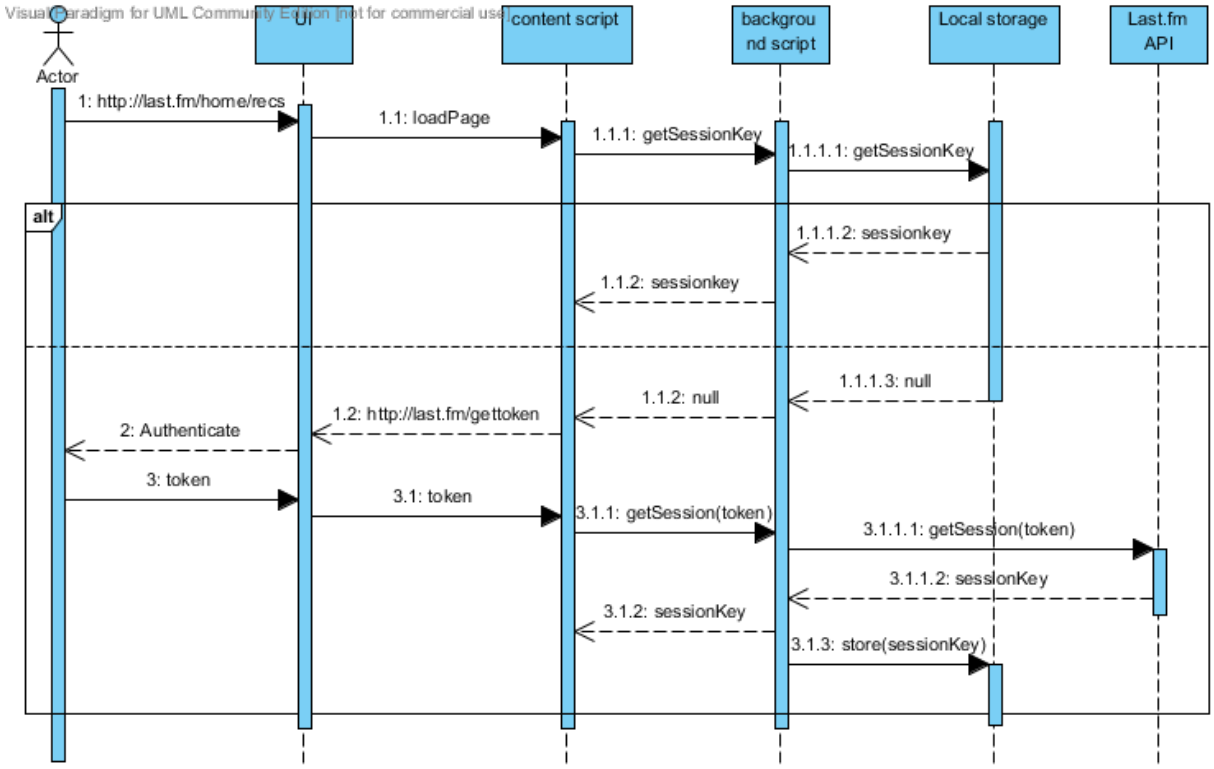


Figure 5.2: Sequence diagram: opening the Last.fm recommendations page part 1: retrieving a session key.

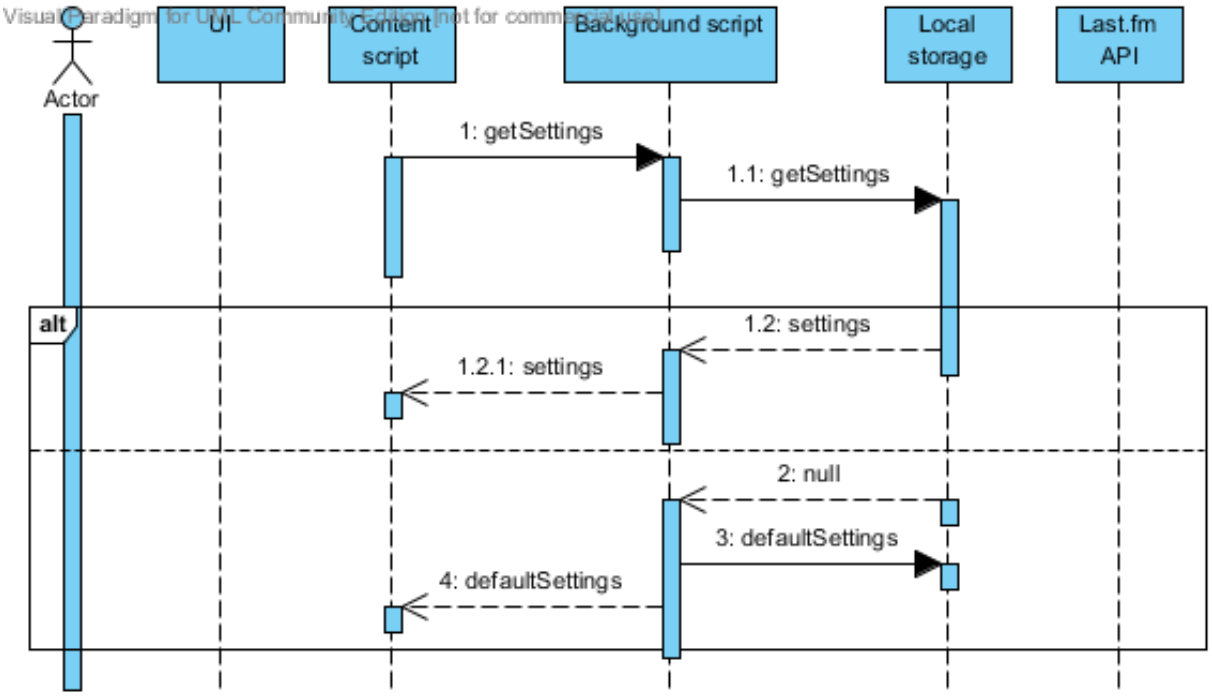


Figure 5.3: Sequence diagram: opening the Last.fm recommendations page part 2: retrieving stored settings.

recommendations R added to the number of top artists T, and the number of neighbours N involved. The resulting time complexity is quadratic, i.e., $O(N^2)$, if A approaches 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. Caching parts of the data structure is possible. However, small changes in the data may have an impact

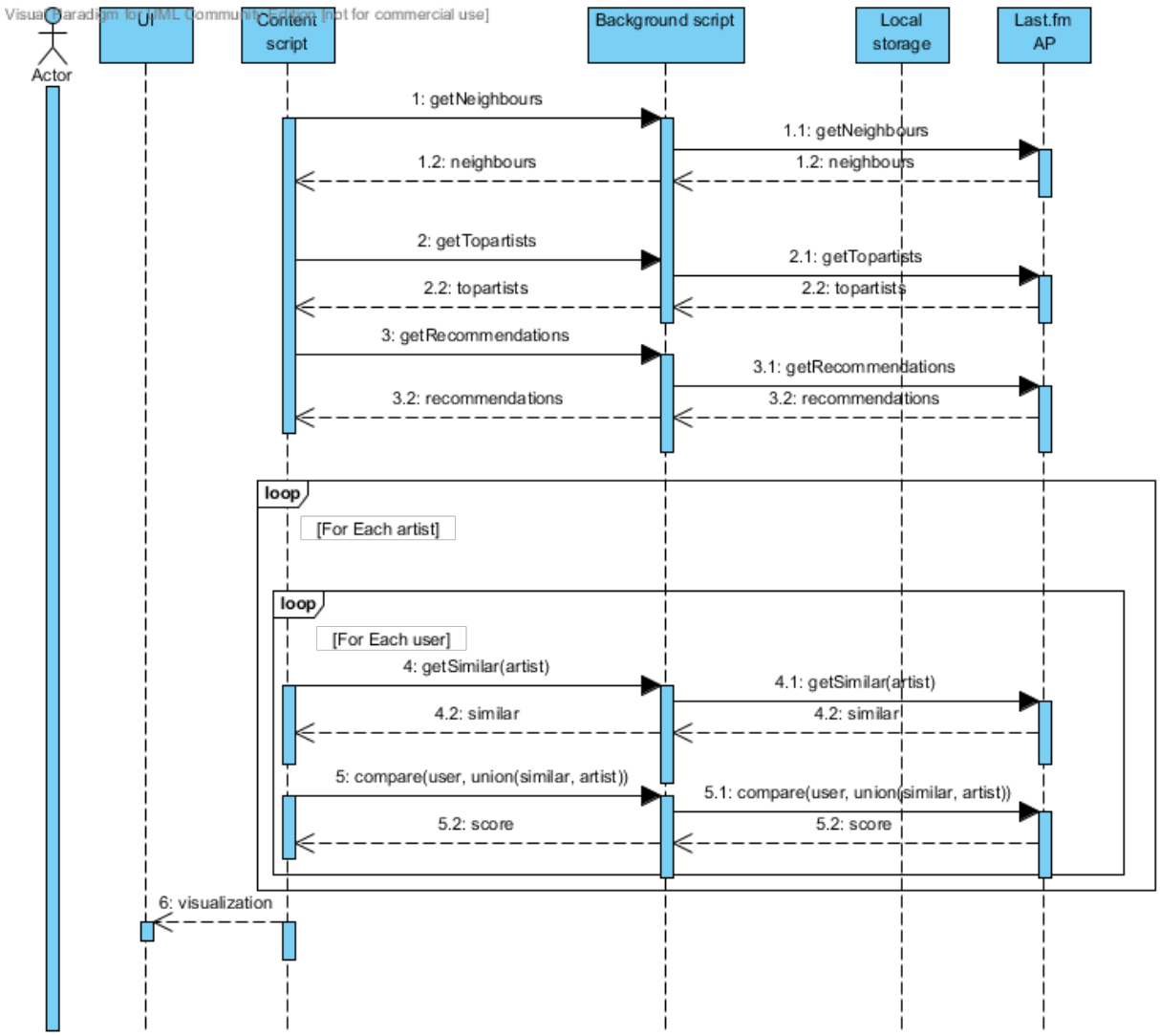


Figure 5.4: Sequence diagram: loading the visualization.

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. This is also a quadratic algorithm. However, its cost is much lower as it requires no expensive calls to the *Last.fm API* in its inner loop.

```
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[6] 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

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.

to support interactions with the user list, which are discussed in the next section. In 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

In the literature we discussed recommender systems and their general context. We listed system properties, described three common recommendation approaches, and listed typical issues and shortcomings of recommender systems. One of these issues is the black box problem for which the end user fails to gain insight into the recommendation process and as a result may have little trust in its recommendations. To solve this problem an explanation system can be used that explains the recommendation rationale.

In the next part of the litature study we looked at a way to visualize this rationale. We came up with a graph-based visualization representing the underlying utility matrix of collaborative recommendation, that uses Holten's edge-bundling algorithm along with node reduction to reduce the number of data dimensions, inspired by a visualization by Valdis Krebs.

Subsequently we looked at an evaluation method for visualization insight developed by Chris North. We also investigated the insight gaining process established by Klein et al. Finally we adapted Ware and Mitchell's visual thinking algorithm to describe how a user would interact with the visualization to solve a certain problem.

A number of visual explanation systems were discussed. To compare these systems we used a number of goals presented by Tintarev and Masthoff.

6.1 Objectives

The first objective described in section 1.1 was to conduct a literature study on techniques for the visualization of music suggestions. This has not been entirely reflected in this text. Nonetheless an effort was made to link presented techniques either to the end product or other examples in the context of music recommendation.

The second objective was to design, implement and evaluate an interactive visualization that will allow the user to gain insight into the recommendation process as well as actively steer the process. The following success criteria for the application were listed in section 1.1.3 of the introduction:

- Aimed at non-expert users with an average to high interest in music;
- Achieve high usability, in particular learnability and memorability;
- Provide transparency.

Although there were some casual listeners among the test users, the majority of participants was representative of the target audience. Results for the perceived usefulness in the last iteration vary between 2 and 5, suggesting that the first criteria has not been met entirely. Still, if the application would be developed further, and more users get tested, the average may still increase.

An overall average SUS score of 80.5 in the final iteration suggests that the usability of the system is good, as perceived by users. However, the learnability of the system has perhaps some room for improvement.

Results indicate that our design can be effective in explaining the rationale of collaborative recommendations. The explanations did not always increase system *trust*, but could give an indication of recommender system bias, as poor recommendations were often not connected to the user's top neighbours. Finally, the explanation system may provide a starting point for further data exploration.

The objective that was not met, was to enable users to actively steer the recommendation process. This is due to the fact that the *Last.fm API* did not support this functionality. Of course an alternative could have been to make use of other methods in the API to construct our own custom recommender system, but we have chosen to explain the artist recommendations made by the actual recommender instead. Another possibility could have been to use another recommender system altogether, but from the systems that were investigated, no significant additional functionality was discovered that could have overcome these issues.

6.2 Future work

6.2.1 Issues

Future work may include addressing problems with visual clutter, and slow data load as listed in table 4.8.

6.2.2 Evaluation

For future user tests, *Last.fm* users could be given a pre-test questionnaire to evaluate the Last.fm recommender and its explanations. Such a benchmark could have proven useful in understanding the usefulness of the application. Other evaluation methods that can be used, are for example expert-based evaluation, and heuristic approaches.

6.2.3 Visualization and music

The focus of the literature study was mainly on providing a context for the elements that were used in the application. To improve the initial design, it might have been better to also incorporate some sort of comparative study of visualization techniques for music, especially if we were to built an explanation system for content-based recommendation. On the other hand, this subject may provide enough content for another thesis.

6.2.4 Extensions

The interactive elements could be enhanced, and the amount of interactive elements increased. For example by allowing interaction with edges, the user could dig deeper into the relationship between artists and the corresponding users.

The visual explanation system could be tested using other data sets and collaborative recommendation systems. The model could be extended for use in a hybrid environment, for example by visualizing also tag-based or other relationships among artists in *Last.fm*.

6.3 Personal reflection

All in all this has been an interesting project. During its course, a lot has been learned and the reasoning on this subject has developed as well. In hindsight there are inevitably things that one would do differently, and this is no exception.

6.3.1 An overview of how the project unfolded

The first months of this project were probably the most difficult ones. It was not always easy to determine what to look for. A lot of papers had to be read again in a later stage, since a lot of details were overlooked due to a lack of context and direction. This is probably typical of students coming from the programme *Schakelprogramma Master Toegepaste Informatica*[1]. Especially in a one year programme, some classes that could have provided background for the thesis subject, may come late in the academic year, such as the course *Gebruikersinterfaces*[2]. As a result, this may have affected the motivation for working on the thesis by the end of the first semester.

During the Christmas break, some papers were reread and a better idea of what needed to be done was formed. As a result, the slow progress in the first semester had to be made undone in the second one. Still, looking back, it is not easy to counter this problem, which is perhaps part of the insight gaining process decribed in this thesis.

6.3.2 Lessons learned

Some things that could have been done differently are probably to have started earlier with user tests. The idea explained in this thesis was developed early on in the project, but was evaluated much later. Conducting user studies early on would have yielded more test results, and provided additional experience. Some issues with the application and testing methods are likely to have been discovered at a much earlier stage as well. One of the reasons for stalling, was lack of confidence in the idea, and also a lack of experience in conducting user studies.

[1] `http://onderwijsaanbod.kuleuven.be/opleidingen/n/SC_50527959.htm`
[2] `http://onderwijsaanbod.kuleuven.be/2012/syllabi/n/H04I2AN.htm`

Bibliography

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

[2] Apple. itunes store sets new record with 25 billion songs sold. URL: `http://www.apple.com/pr/library/2013/02/06iTunes-Store-Sets-New-Record-with-25-Billion-Songs-Sold.html`, 2013. [Online; accessed 01-June-2013].

[3] Bandcamp. Artists — bandcamp. URL: `https://bandcamp.com/artists`, 2013. [Online; accessed 01-June-2013].

[4] 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.

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

[6] M. Bostock. Hierarchical edge bundling. URL: `http://bl.ocks.org/mbostock/1044242`, 2013. [Online; accessed 05-June-2013].

[7] J. Brooke. Sus - a quick and dirty usability scale. URL: `http://hell.meiert.org/core/pdf/sus.pdf`, 1996. [Online; accessed 20-March-2013].

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

[9] 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].

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

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

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

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

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

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

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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].

[23] 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.

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

[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] Y. Song, S. Dixon, and M. Pearce. "a survey of music recommendation systems and future perspectives". URL: `http://www.eecs.qmul.ac.uk/~yadings/papers/song2012a.pdf`, 2012. [Online; accessed 02-June-2013].

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

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

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

[53] 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.

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

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

[56] K. Verbert, H. Drachsler, N. Manouselis, M. Wolpers, R. Vuorikari, and E. Duval. Dataset-driven research for improving recommender systems for learning. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, LAK '11, pages 44–53, New York, NY, USA, 2011. ACM.

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

[58] Wikipedia. Last.fm - wikipedia, the free encyclopedia. URL: `http://en.wikipedia.org/wiki/Last.fm#Recommendations`, 2013. [Online; accessed 26-November-2012].

[59] World Wide Web Consortium (W3C). Implementations - svg. URL: `http://www.w3.org/Graphics/SVG/WG/wiki/Implementations`, 2010. [Online; accessed 26-December-2012].

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

[61] 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.

[62] 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.

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

Task lists for the user tests

B.1 Task list 1: testing insight and usability

First 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. Next tasks B.1, B.2 and B.3 are performed.

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

Tables B.4, B.5, and B.6 give an overview of the tasks used in the user study to evaluate the settings menu.

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

Tasks B.7, B.8 B.9, and B.10 are used to further evaluate the explanation system properties.

Table B.1: Task 1.1: hypothesis generation, no interaction allowed.

Goal/Output	Getting an idea of the user's mental model about the visualization when he/she is not allowed to interact with it.
Inputs	The user has an account, and has built up some listening history.
Assumptions	The user is logged in. The data has loaded.
Steps	The user will try to get an overview of the displayed data. Through eye-movements he/she will explore the visualization. The user forms a hypothesis on the visualization rationale.
Estimated time	5 to 10 minutes.
Instructions	Answer the following questions without interacting with the visualization: 1. Describe what you see. Which visual elements stand out? Which general structures can be identified? 2. What do you think the visualization does? 3. Which the elements of the user interface, do you think allow interaction? 4. 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?

Table B.2: Task 1.2: Further familiarization, interaction allowed.

Goal/Output	Getting an idea of the user's mental model about the visualization.
Inputs	See table B.1.
Assumptions	See table B.1.
Steps	The user verifies his/her initial mental model through interactions with the visualization. If the initial mental model is not confirmed, it is adjusted.
Estimated time	
Instructions	Try to interact with the visualization. Answer the following questions: 1. Which of the artists displayed in the graph are artist suggestions? 2. What are the links or edges in the visualization? 3. Suppose you want to add an item to your profile, what steps would you undertake?

Table B.3: Task 1.3: Adding an artist to the music library and motivating the choice(s) made.

Goal/Output	
Inputs	See table B.1.
Assumptions	See table B.1.
Steps	The user clicks an artist of his/her choice. The user clicks *Add to library* and confirms his/her action. The item is added to the profile and the visualization refreshes.
Estimated time	1 to 5 minutes.
Instructions	Add an item to your profile. Answer the following questions: 1. Why did you choose that particular item? 2. Can you give any other reasons why you should pick this item? 3. Can you give reasons for choosing one of the other items?

Table B.4: Task 2.1: Change the number of shown recommendations up to 20.

Goal/Output	The number of displayed recommendations in the graph has changed.
Inputs	The user has a *Last.fm* account. The user has authorized the application.
Assumptions	The user is logged in. The user has navigated to the recommendations page and the vsiualization has loaded.
Steps	Click the *Settings* button and alter the slider for the number of recommendations. Click *Save*.
Estimated time	Less than a minute.
Instructions	Change the number of shown recommendations up to 20.

Table B.5: Task 2.2: Change the threshold to 0.3.

Goal/Output	The threshold used to cluster items has changed, which will affect the connectivity of the graph.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	Click the *Settings* button and alter the slider for the threshold. Click *Save*.
Estimated time	Less than a minute.
Instructions	Change the threshold to 0.3.

Table B.6: Task 2.3: Change the colours to an encoding that you like.

Goal/Output	The colour encodings for hover and click actions has changed, as well as the colour of the active user profile.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	Click the *Settings* button and alter the the colour settings. Click *Save*.
Estimated time	About a minute or more.
Instructions	Change the colours to an encoding that you like.

Table B.7: Task 3.1: Find three neighbours that are closely related to you, based on the visualization.

Goal/Output	The user can find three closely related neighbours and can give an adequate motivation for his/her choice.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	Based on the visual thinking algorithm in table 2.1.
Estimated time	3 minutes or more, depending on past experience.
Instructions	Find three neighbours that are closely related to you, based on the visualization. Explain why these are closer neighbours than others neighbours on the graph.

Table B.8: Task 3.2: Find three recommended artists you think are interesting.

Goal/Output	The user can find three interesting artist recommendations and give an adequate motivation for his/her choice.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	Based on the visual thinking algorithm in table 2.1.
Estimated time	3 minutes or more, depending on past experience.
Instructions	Find three recommended artists you think are interesting. Explain why these artists are more interesting than other recommendations shown in the graph.

Table B.9: Task 3.3: Explain the recommendation rationale (transparency).

Goal/Output	The high level algorithm for collaborative filtering.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	Based on the visual thinking algorithm in table 2.1.
Estimated time	3 minutes or more, depending on past experience.
Instructions	Explain the recommendation rationale. How do you think *Last.fm*'s recommender system works?

Table B.10: Task 3.4: Find a suggestion for an artist you didn't know about.

Goal/Output	A new suggestion.
Inputs	See table B.4.
Assumptions	See table B.4.
Steps	The user looks at each of the suggestions and points out those that are new to him/her.
Estimated time	About a minute to find new suggestions. Investigating interesting recommendations and answering the questions may take up up to 15 minutes.
Instructions	Find a suggestion for an artist you didn't know about, and answer the following questions: • 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 C

Quantified self

Figure C.1 and table C.1 give an overview of the activities for thsis thesis based on *Toggl*[1] entries. Table C.2 shows the statistics for social interactions related to the thesis, including tweets using the #thesis12 hash tag on *Twitter*[2], posts on the *SoundSuggest* blog[3] and comments on blogs by peers, and read papers and suggestions on *Tinyarm*[4].

[1] `https://www.toggl.com/`
[2] `https://www.twitter.com/`
[3] `http://soundsuggest.wordpress.com/`
[4] `http://atinyarm.appspot.com/`

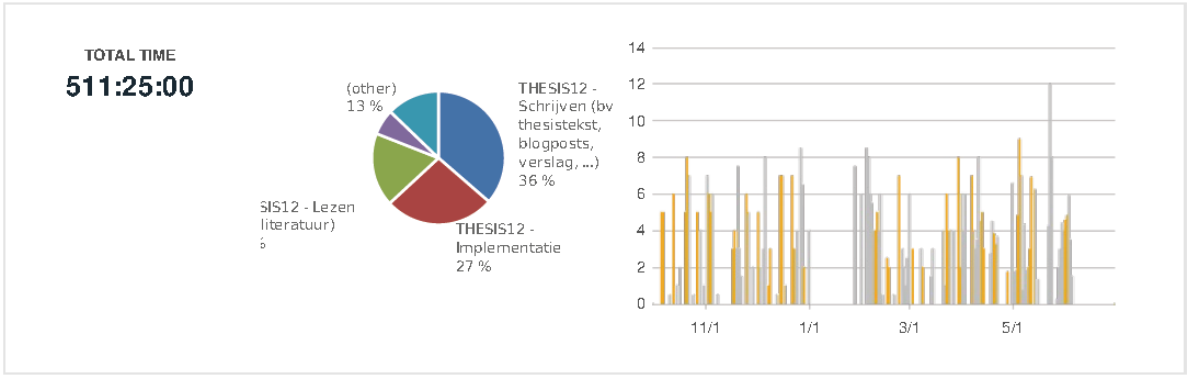


Figure C.1: Graphs generated by *Toggl*.

Table C.1: Approximation of the distribution of activities for this thesis, based on *Toggl* entries.

Activity	**Time**
Schrijven (bv. thesistekst, blogposts, verslag, ...)	186:00:00
Implementation	136:50:00
Lezen (bv. literatuur)	91:10:00
Evaluatie	31:50:00
Presentatie (bv. voorbereiding, geven)	17:00:00
Reflectie (bv opstellen planning, interpretatie resultaten, etc.)	13:30:00
Ontwerp	12:30:00
Social Network Actvities (bv.twitter, comments op andere blogs, suggesting papers met tinyarm, etc)	11:20:00
Meetings (bv. met begeleider, mede-studenten, etc.)	11:15:00
Total time	511:25:00

Table C.2: Approximation of the distribution of activities for this thesis, based on *Toggl* entries.

Tool	**Unit**	**Quantity**
Twitter	Tweets *#thesis12*	∓66
Wordpress	Posts	36
	Pages	15
	Comments	±30
Tinyarm	Papers (read)	45
	Recommended	3

Appendix D

Scientific article

An explanation system for collaborative music recommendation using graph-based visualization

Joris Schelfaut *

Faculty of Computer Science, Katholieke Universiteit Leuven, Belgium

ABSTRACT

The rationale of recommender systems is often opaque towards the end user, possibly causing decreased levels of acceptance of its recommendations. Explanation systems can overcome this problem by providing insight into the reasoning behind suggestions[7].

In this paper we will look at a white box model for collaborative filtering. This model is implemented as a visual explanation system called *SoundSuggest* which aims to explain Last.fm's collaborative recommender. The system is evaluated through a user study. We will investigate the quality of insight gaining and its effects on trust, effectiveness and persuasion of Last.fm's recommendations.

Keywords: *recommender system, insight gaining, interactive visualization, usability*

Table 1: Explanation aims. Table adapted from Tintarev and Masthoff [21].

Aim	Definition
Transparency (Tra.)	Explain how the system works.
Scrutability (Scr.)	Allow users to tell the system is wrong.
Trust	Increase users' confidence in the system.
Effectiveness (Efk.)	Help users make good decisions.
Persuasiveness (Pers.)	Convince users to try or buy.
Efficiency (Efc.)	Help users make decisions faster.
Satisfaction (Sat.)	Increase the ease of usability or enjoyment.

1 INTRODUCTION

Music catalogues for online retail have become immense over the past decades. Well-known artists and tracks make up a very small portion of this item space, which is known as the *Long-tail phenomenon*. As a result, finding new, interesting music has become a challenging task. *Recommender systems* try alleviate this problem by filtering the item repository based on a user's music taste. Taste can be modelled by analyzing user preferences and tracking user behaviour, e.g., by analyzing a user's listening history[19].

There are two commonly applied filtering strategies[16]:

- **Content-based filtering (CBF)**: Using chosen or modelled features of items to define similarity between items in the user profile and candidate suggestions;
- **Collaborative filtering (CF)**: Using overlap of item sets of each user profile to find possible suggestions in the difference of these item sets.

CF-based approaches, or hybrid approaches of CBF, CF and possibly other strategies, are often applied in music recommendation. Although these recommender systems have proven to be successful in terms of prediction accuracy, the success of recommender system also relies on the trust in its recommendations by the end user. If the user does not know why a particular item is recommended to him, the user may be reluctant to check it out. Herlocker et al. [7] describe this issue as the *black box problem*. To improve acceptance of recommendations, they propose to build an explanation system presenting the user with a *white box model* of the recommender system rationale.

This paper looks at an explanation system for collaborative music recommendation that uses a graph-based visualization. The explanation system will be evaluated based on seven aims described by Tintarev and Masthoff [21] listed in table 1. Also learnability and memorability, properties of usability as described by Nielsen[12], are also evaluated. An insight evaluation method developed by Chris North [14] is used to measure transparency. Usability evaluation methods are used to measure satisfaction, efficiency, learnability (Learn.) and memorability (Mem.). Trust, effectiveness, and persuasiveness are also evaluated during the user study. Scrutability is not supported by the explanation system.

*Joris Schelfaut - Louvain, Belgium, E-mail: joris.schelfaut@student.kuleuven.be

2 RELATED WORK AND BACKGROUND

This paper draws from concepts from the field of recommender systems, insight gaining and visualization. We will also look at the number of explanation systems that have been described in scientific literature and compare them based on the aims listed in table 1.

2.1 Collaborative filtering

Recommender system data is usually represented in the form of a matrix in which users correspond to rows, and items correspond to columns. This matrix is often referred to as the *utility matrix*. An entry $a_{i,j}$ in this matrix corresponds to a quantification of preference of user i for item j. The goal of the recommendation algorithm is to find an estimation for the blank entries in the matrix [16].

Often the utility matrix is very sparse. For systems with thousands of users and items, users will generally only have rated a small subset of those items. The problem raises significant performance issues for new users, as they have few items in their rating history, or new items, as few people have that particular item in their rating history. This problem is often referred to as the *cold start problem* [7, 16].

Another issue that is typically related to collaborative filtering, is the *gray sheep problem*. This phenomenon occurs when a user profile has no or very few other similar users associated with it. This makes it hard to establish a true 'neighbourhood' for this user[24].

2.2 Insight gaining

In [14] 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. Instead, insight is considered a multidimensional property; it is complex, deep, qualitative, unexpected, and relevant[14, 23].

The quality of insight can then be determined by quantifying each of these characteristics[14]. North describes methods to evaluate insight gaining through visualizations, such as usability testing, heuristic evaluation, cognitive evaluation, and controlled experiments on benchmark tasks[14].

Chris North points out that controlled experiments suffer from problems that may hinder effective evaluation of previously listed characteristics of insight. For example the predefined nature of such experiments may decrease the amount of unexpected insight. Instead he prefers an evaluation method based on an open-ended protocol, qualitive insight analysis, and an emphasis on domain relevance[14].

Yi et al. [23] identify four processes, that are often intertwined, through which insight is established. The insight gaining processes are provide overview, adjust, detect pattern, and match mental model.

2.3 Visualization

Munzner et al. [17] identify limitations in computational and cognitive performance, and screen size for visualizing data on a screen. To alleviate these problems a wide range of visualization techniques have been developed. An overview of such techniques can be found in [10], [22], and [8].

Clutter and data overload are two problems that are common in information visualization[17]. Examples of clutter, data, and dimensionality reduction techniques are spatial distortion, clustering, change in opacity, and edge-bundling[4, 8, 9].

To describe how users interact with visualizations, Ware and Mitchell [22] list a number of *visual thinking algorithms*. A visual thinking combines perceptual and cognitive actions into a process, as the user interacts with the visualization and explores the data space[22].

2.4 Explanation systems

A number of explanation systems have been developed for recommender systems. In [15] an application called *PeerChooser* is presented by O'Donovan et al. It uses a graph-based visual explanation system for CF. Interactive elements incorporated in the visualization allow the active user to manipulate his/her neighbourhood. The *SmallWorlds* application by Gretarsson et al. [6] uses a similar approach. *Pharos* [24] also builds on ideas brought forth in [7] and [15]. The application computes a social map from the user's behaviour in content-based websites. The *TasteWeights* application by Bostandjiev et al. [1] is created for a hybrid recommendation system. It uses a graph-based approach to visualize relationships between the different recommender algorithms [1]. *SFViz* was developed by Gou et al. [5] and uses a visualization of a tag-based network to find friends based on mutual tastes in music.

Table 2 shows which of the characteristics described by Tintarev and Masthoff, were pursued for each of the explanation systems.

3 VISUALIZATION DESIGN

3.1 Translating the recommender rationale

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 $U \cap I = \emptyset \wedge E \subseteq U \times I$[3]. Each non-blank entry in the utility matrix will then correspond to an edge. Figure 1 shows how a matrix is transformed into a dual graph.

Table 2: A comparison of the visual explanation systems, based on the aims by Tintarev and Masthoff listed in [21].

	Tra.	Scr.	Trust	Efk.	Pers.	Efc.	Sat.
PeerChooser	x	x		x			x
Pharos	x		x				
SFVis	x	x					
SmallWorlds	x	x		x			x
TasteWeights	x	x	x	x		x	

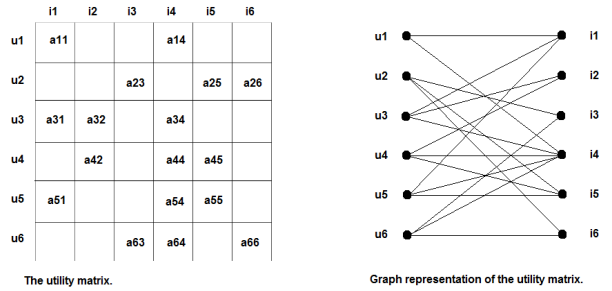


Fig. 1: Transforming the utility matrix into a dual graph: two distinct sets of nodes, users and items, only share edges between nodes of different sets.

The set of nodes U corresponds to the set of users, and the 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.

3.2 Data and dimensionality reduction

Based on a visualization design by Valdis Krebs [20], a dimensionality reduction can be performed on the dual graph through *row reduction*. One set of nodes is eliminated from the graph and is represented as implicit information in the edges. Figure 2 shows an example of this idea.

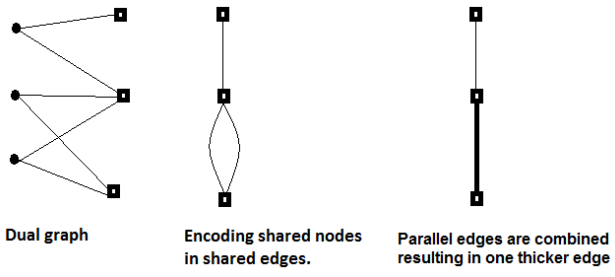


Fig. 2: A row reduction operation on each pair of edges in a dual graph will result in a dimensionality reduction where one set of nodes is removed from the graph. Additional data reduction can be achieved by clustering edges into a thicker edge. Edge thickness then depends on the number of edges involved.

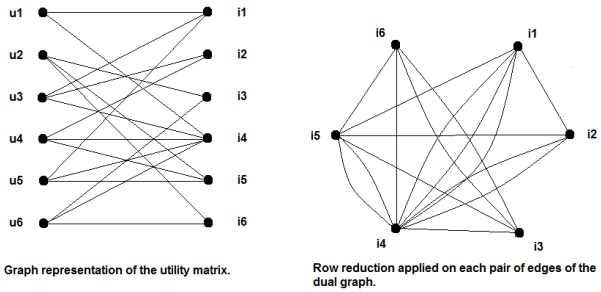


Fig. 3: Row reduction applied on the graph in figure 1.

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 occur in neighbouring profiles. Figure 3 shows how the dual graph from figure 1 is transformed into a circular graph layout with the remaining item nodes.

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.

3.3 Retaining contextual information

An important remark made in [11] 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. Therefore, 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 shown in figure 4, 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.

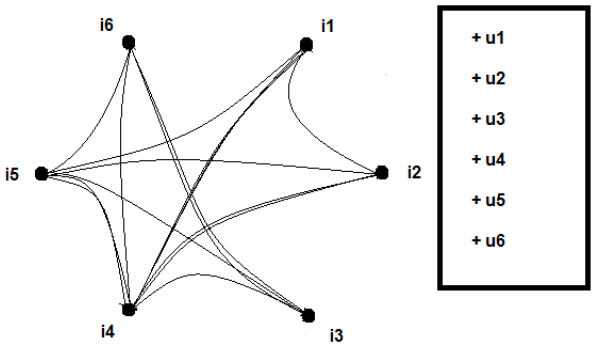


Fig. 4: The resulting visualization serving as a white box model for collaborative filtering.

4 EVALUATION

A user study was conducted to evaluate the white box model presented in section 3. In this study the explanation system aims listed in table 1 are evaluated.

4.1 Methodology

Transparency is tested by evaluating insight into the recommendation process based on North's evaluation method. We will use the think aloud protocol to obtain observational data. In particular we are looking for a user to make "domain specific inferences and hypotheses"[14].

Satisfaction, efficiency, and learnability are tested through think aloud usability testing and a summative *system usabiliy scale* (SUS) questionnaire. SUS is a *Likert scale* method consisting out of 10 questions, listed in figure 3, to investigate the subjective usability of an application[2]. Memorability is tested by asking test users that participated in previous iterations to explain the recommender rationale again at the beginning of the test.

Trust, persuasiveness, and effectiveness are evaluated through direct feedback from the test subjects.

4.2 Participants

Test users were selected from the campus and among acquaintances and were between 21 and 26 years of age. For all of the iterations combined, a total of 15 users participated in the user study of whom 12 were male and 3 were female. All users had an interest in music and listened actively to music at least once a day. Although they had a notion of what a recommender system was, none of them knew how recommendation algorithms worked.

Nielsen [13] proposes to make iterations short by limiting the number of test users. He argues that five test users per iteration are enough to find most of the usability issues. By iterating and addressing surfaced usability issues between each iteration, after a couple of iterations all usability problems will have been resolved.

Table 3: System usability scale questions.

Q1	I think that I would like to use this system frequently.
Q2	I found the system unnecessarily complex.
Q3	I thought the system was easy to use.
Q4	I think that I would need the support of a technical person to be able to use this system.
Q5	I found the various functions in this system were well integrated.
Q6	I thought there was too much inconsistency in this system.
Q7	I would imagine that most people would learn to use this system very quickly.
Q8	I found the system very cumbersome to use.
Q9	I felt very confident using the system.
Q10	I needed to learn a lot of things before I could get going with this system.

Table 4: The distribution of test users used in the evaluations for each iteration.

	Iteration			
	1	2	3	4
Number of users	5	5	5	10
From previous iterations	-	2	3	5

As a result, the test users were spread among four different iterations. Some of the users participated in multiple tests to provide direct feedback on changes made between the iterations. The distribution of users can be seen in table 4.

4.3 Prototyping

Over the four iterations the application was incrementally improved. Table 5 gives an overview of which aims were evaluated for the prototype in each iteration.

The first prototype, shown in figure 6, was made out of paper. *Paper prototypes* are relatively easy and cheap to make. Evaluation of this kind of prototype allows to detect usability problems at an early stage in the development process at a low cost. This avoids having to fix these usability problems in an actual im-

Table 5: The explanation aims that were evaluated in each iteration.

Aim	**Iteration**			
	1	2	3	4
Tra. Sat., Efc., Learn.	x	x	x	x
Mem.		x	x	x
Trust, Efk., Pers.				x

plementation, which is much more expensive[18].

The second prototype, displayed in figure 7, is an implementation of the visualization using the `D3` *JavaScript* library. This version uses the same static data as in the paper prototype. In addition to the aims listed in table 5, the success of the conversion from paper to digital prototype is evaluated as well.

The third prototype uses the visualization in a Chrome browser extension that is injected into the *Last.fm* recommendations page[1]. The visualization uses live data from the logged in user. This, in combination with the context of the Last.fm website, increases the domain relevance for insight evaluation. The main objective of the user study for this prototype was to find usability issues before moving on to a test with more users.

In the final prototype all of the other aims were evaluated, apart from scrutability, as this was not supported by Last.fm's API[2]. An example of the visualization is shown in figure 8.

4.4 Results

4.4.1Transparency

All test users were able to derive the recommendation rationale from the visualization. However, there were some differences in the speed of the insight gaining process.

The test was designed to better distinguish between the steps of the insight gaining process. To simulate the first step, *provide overview*, users were asked to form an initial mental model before interacting with the visualization. For all iterations most users saw the edges as content-based relationships, e.g. artists are connected based on genre. Only two users managed to get the visualization rationale right the first time. Based on the visualization they were able to describe the recommendation rationale.

Adjust, *detect pattern*, and *match mental model* were simulated in the next part of the test by allowing interaction with the visualization. Users could now dig deeper into the data model, gaining understanding about relationships among its elements. It should be noted that over the iterations, changes in the amount of data displayed, as well as changes in the graph's layout had an influence on the insight gaining speed.

In the first iteration, see figure 6, parallel edges were clearly visible, whereas in the other iterations, see for example figure 7, Holten's edge-bundling algorithm made parallel edges overlap. Test users indicated that this made it harder to see the link between the number of edges between an artist, and the number of highlighted neighbours. This was also supported by observational data, as this kind of stories did no longer occur in think aloud tests, unless insight into visualization's rationale had already been gained via another way.

As the number of data elements displayed increased, comparing prototypes in figures 7 and 8, it became harder to gain insight, as the total amount of edges made it harder to distinguish between them. This is probably to be expected, as small graphs are easier to analyze[8].

Usually the visualization rationale learned when the user saw that a user profile corresponded to set of interconnected nodes. By comparing overlaps between artist node sets and the highlighted active profile, they understood how neighbouring profiles were interrelated, and how recommendations were computed.

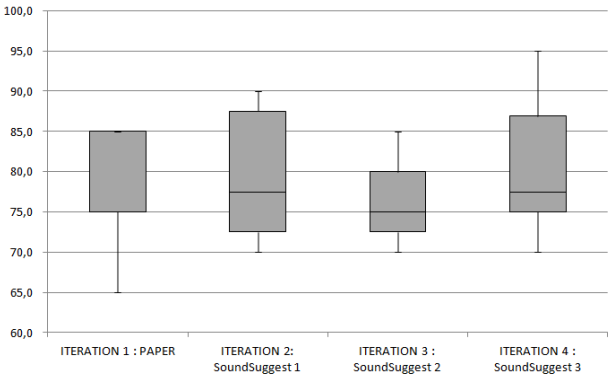


Fig. 5: The SUS results for each iteration, visualized as box plots.

4.4.2Satisfaction, efficiency, learnability and memorability

To get an idea of the perceived usability, the SUS questionnaire results are used. These changed over the iterations: new users were introduced and the design changed over time. Also, when comparing the first iteration with the others, the interaction with a paper prototype is still rather different than working with a digital version. The results are shown in figure 5.

Figures 9 to 12 show the distribution of answers for each iteration. The questions with the lowest scores overall were the first question and the seventh question.

Low scores on `Q1` may point to low satisfaction, but at the same time there were very positive remarks as well. Another explanation might be that low scores correspond to users that simply don't use Last.fm frequently. Based on the results for `Q3`, the majority of test users indicated that they thought the application was easy to use.

Although there were no negative votes for `Q7`, a lot of users were not convinced that using the application was easy to learn. Although the results for `Q10` suggest otherwise based on iteration 2 and 4, this question might be more prone to possible bias introduced dur-

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

ing the test, which is perhaps confirmed by results for `Q4` in iteration 1 and 3.

Even though the application did not stay exactly the same over the iterations, test users that took part in two or more user tests were able to apply their experience from the previous test to the new test. Recalling the recommender rationale sometimes required some familiarization with the application again.

4.4.3Persuasion, effectiveness, and trust

In terms of trust, four scenarios were investigated: effective/bad, known/new recommendations.

It turns out the Last.fm recommender has some bias towards certain clusters of artists. Artists from a smaller music scene tend to be affected by regional effects. For example according to Last.fm, a similar artist to *De Kreuners*, a Belgian rock band, is *Samson & Gert*, a performer of music targeted to children. Another example is that bands that have certain musicians in common often are considered similar, even though their music styles are not. Usually test users already knew about these artists and indicated that this decreased their trust in the recommender system.

Similarly, bad recommendations that were new to the user also decreased trust in the recommendations.

Not all test users received this kind of biased recommendations and effective recommendations would increase the user's trust in the recommender system. Good new recommendations would increase their trust in the system more than artists they already knew about.

In some cases the explanation system helped to identify bad recommendations, as the active user's top neighbours would not have these items in their profile.

As users gained insight in the visualization and the recommender system behind it, the user's trust in the system increased as well. Persuasion was harder to measure. Typically test users would look for artist nodes where a lot of edges originated from, or users with a high similarity score.

Although the explanation system was not always as effective in helping to find good recommendations, it provided an additional means for the user to establish his/her own approach for finding recommendations. For example a user would look at neighbours for artist suggestions, rather than just the artist recommendations by Last.fm.

5 CONCLUSION AND FUTURE WORK

This paper has described a visual explanation system collaborative recommendation. The design was evaluated through user studies, first as a paper prototype which was later implemented as a Chrome Extension for the Last.fm website to explain its music recommendations. We used aims proposed by Tintarev and Masthoff, with additional usability metrics listed by Nielsen to evaluate these prototypes.

Results indicate that our design can be effective in explaining the rationale of collaborative recommendations. However, the learnability of the system still has some room for improvement. An overall SUS score in the final iteration of 80.5 suggests that the usability of the system is good, as perceived by users. Finally, the explanation system can help increase trust in the recommender system and may provide a starting point for further data exploration.

To improve the application, issues such as data density, slow data loads should be addressed further. It would also be interesting to see how the explanation system would perform for another collaborative recommender system.

REFERENCES

[1] 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.

[2] J. Brooke. Sus - a quick and dirty usability scale. URL: `http://hell.meiert.org/core/pdf/sus.pdf`, 1996. [Online; accessed 20-March-2013].

[3] 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].

[4] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, Nov. 2007.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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].

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

[15] 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.

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

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

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

[19] Y. Song, S. Dixon, and M. Pearce. "a survey of music recommendation systems and future perspectives". URL: `http://www.eecs.qmul.ac.uk/~yadings/papers/song2012a.pdf`, 2012. [Online; accessed 02-June-2013].

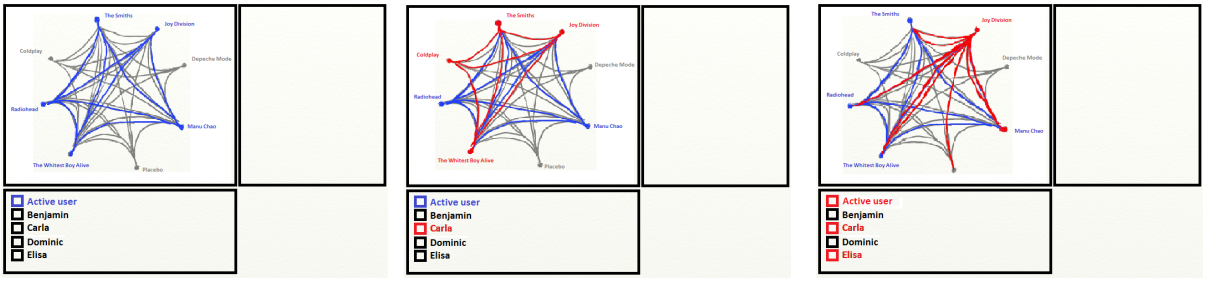
[20] J. Steele and N. Iliinsky. *Beautiful Visualization: Looking at Data through the Eyes of Experts*. O'Reilly Media, Inc., 1st edition, 2010.

[21] 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.

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

[23] 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.

[24] 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.

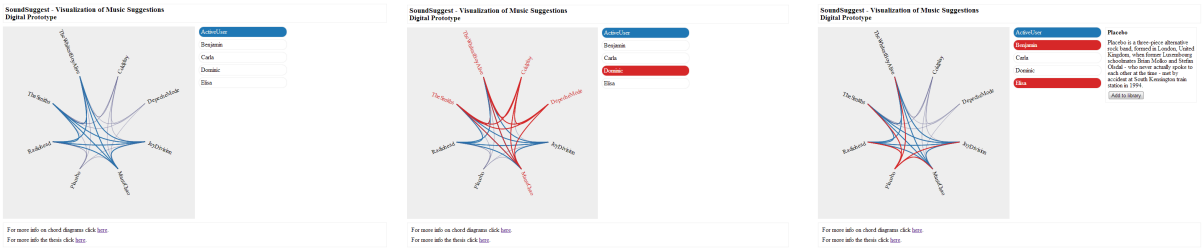


(a) The default visualization with the active user's items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Fig. 6: A selection of the screens used in the user study with paper prototype.

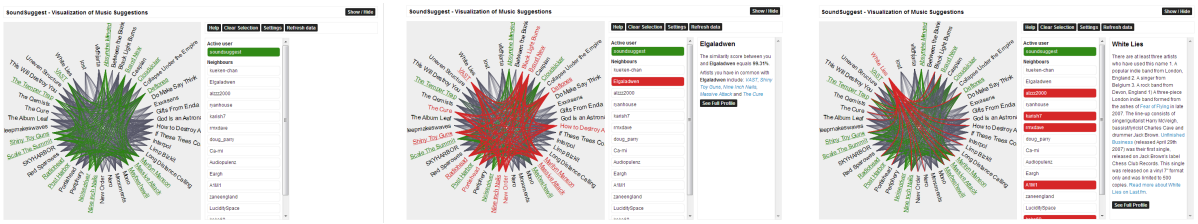


(a) The default visualization with the active user's items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Fig. 7: A selection of the screens used in the user study with the first digital prototype.



(a) The default visualization with the active user's items highlighted.

(b) Clicking a user list element.

(c) Clicking an item node.

Fig. 8: A selection of the screens used in the user study with the third digital prototype.

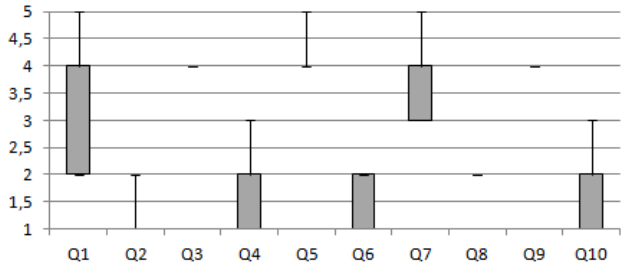


Fig. 9: The SUS results for each question for iteration 1.

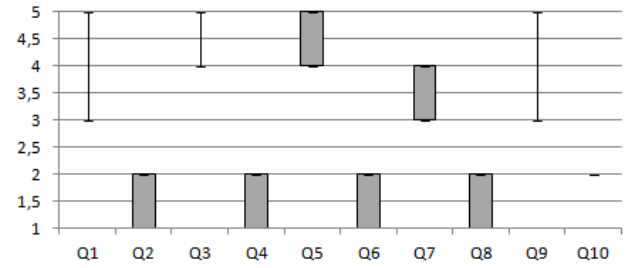


Fig. 10: The SUS results for each question for iteration 2.

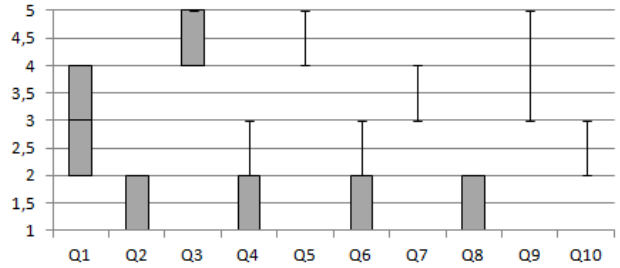


Fig. 11: The SUS results for each question for iteration 3.

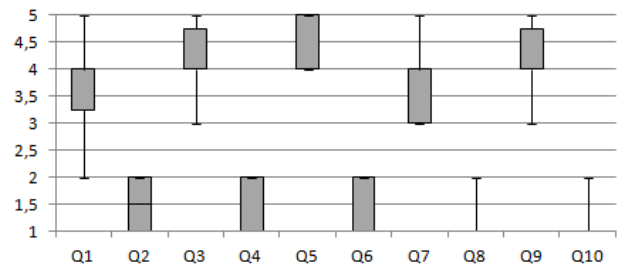


Fig. 12: The SUS results for each question for iteration 4.

Appendix E

Poster

Visualization of music suggestions

KATHOLIEKE UNIVERSITEIT LEUVEN

FACULTEIT WETENSCHAPPEN

Master
Toegepaste Informatica

Masterproef
Joris Schelfaut

Promotor
Erik Duval,
Joris Klerkx,
Katrien Verbert

Academiejaar
2012-2013

Context

The **rationale** behind the generation of suggestions **by recommender systems** is often **not transparent** towards the end user.

Many recommenders exist, for example **music recommenders** like *Last.fm*, *Grooveshark* and *Spotify*.

The user can gain insight into this rationale through an **explanation system**.

Visualizations offload cognitive load and often are an effective way to support explanation.

Visual explanation systems have been developed to help users gain insight

Objective

- Create a visual explanation system for a music recommender system.
- Test the application through a series of user tests

Results

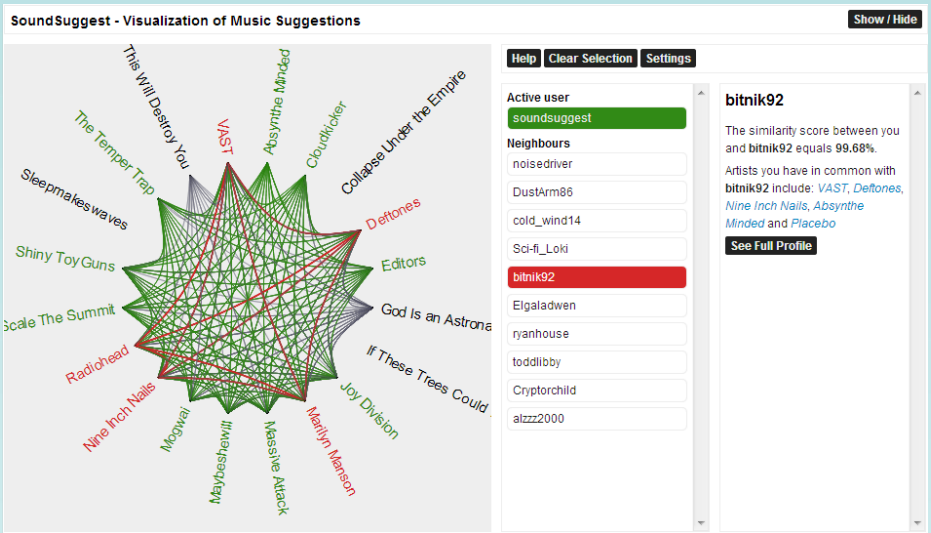
An explanation system for Last.fm recommendations: **SoundSuggest**

Tested through user tests
- **Think aloud protocol**
- **SUS** questionnaires

Does the application help gaining insight into the recommendation algorithm?

→ *Yes, test users were able to reconstruct the general recommendation rationale*

SoundSuggest



- HTML, SVG, CSS, JavaScript
- Chrome extension
- Data-driven documents (D3.js)
- Last.fm API

Appendix F

Source code

The source code can be found on the disk added to this text.

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