KU LEUVEN

FACULTY OF SCIENCE

Visualization of music suggestions

A visual explanation system for collaborative filtering

Joris SCHELFAUT

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

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

Mentor: *(J. Klerkx)*
Affiliation *(optional)*

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

Academic year 2012-2013

Contents

Chapter 1

Introduction

1.1 Music suggestions and explanation systems

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

In their essence, recommender systems can be seen as filters applied on a large data sets. Ever since computer engineers started to develop this kind of systems, a wide range of algorithms have been designed and implemented to compute item recommendations[5, 37, 44, 45]; each of them with their own advantages and disadvantages.

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

To solve this problem, you will need some kind of explanation system that provides a reasoning to arrive at the results. An ambitious approach would be to explain each step of the recommendation algorithm, but this not always possible or desired. Indicating which of your tracks are closely related to the given recommendations, the system's confidence in the accuracy of the suggestions, et cetera, also help giving some additional context in explaining why a particular recommendation would be interesting[18]. 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[3, 6, 9, 16, 17, 43].

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.

1.2 Thesis objective

The goal of this thesis is to design, implement and evaluate visualization and interaction techniques that will allow the user to gain insight into the recommendation process as well as actively steer the process. The elaboration of this thesis consists out of a literature study on the topic of visualization of music suggestions, and secondly a similar application that is designed and implemented[32].

In the design of the application findings of the literature study and comparative study of visual explanation systems are taken into account. From this initial design a hypothesis and expected result are derived. This hypothesis is tested through user tests. If possible, the application is improved. This process can be repeated, incrementally improving the application.

The scenario in section 1.1 sketches the context in which this application can be used.

1.3 A multi-focal perspective

Although the focus of this thesis lies on the creation of a visual explanation system for music recommendation, this thesis covers aspects of several research domains. Therefore, this text can be viewed from a multi-focal perspective. The various facets that can be distinguished are as follows: item recommendation, which is in turn a subfield of data mining, insight gaining and sensemaking, and information visualization and visual data mining. Also the graph drawing problem is an important topic in this paper. And last but not least, human-computer interaction can be seen as the overarching concept.

To make the forementioned topics more concrete in the context of this thesis, the following subsections try to explain how each of these topics is relevant, as well as explain how they are interrelated.

1.3.1 Item recommendation

Recommender systems form one of the major topics of this thesis. In particular this thesis tries to address the black box problem. In a paper by Herlocker et al. [18], recommender systems are compared to a black box. This black box generates item recommendations seemingly at random, leaving the user guessing as to how recommendations were computed. Herlocker et al. propose to create an explanation system, i.e., the white box, to overcome this black box problem.

Although there are many variations of recommendation algorithms, the focus of this thesis is on collaborative filtering-based (CF) item recommendation. This algorithm is studied in the context of music recommendation. The CF algorithm tries to establish similarities between user profiles, in this case music libraries with additional listening history and/or item ratings. Items in the difference between items sets of similar profiles are then candidate recommendations[45].

1.3.2 Insight gaining and sensemaking

Another aspect of this thesis is related to insight gaining. In order to develop an explanation system, it is interesting to explore how a user arrives at insight. We will try to define the insight gaining process and the related concept of sensemaking. Sensemaking is not

uniquely defined as its meaning may slightly vary depending on the context[63]. In short, sensemaking can be explained as the effort by an individual to understand the underlying information structures with the objective to answer task-specific questions[46, 63].

1.3.3 Information visualization and visual data mining

As the explanation system will be using visual elements, concepts from the field of information visualization will be used. We will look at general characteristics of information visualization and interactive visualization. Also, various visualization techniques, and clutter reduction techniques are discussed.

1.3.4 Graph drawing

To determine whether or not graphs are applicable to visualize a concept, Herman et al. [20] pose the following question: "is there an inherent relation among the data elements to be visualized?" As the structure behind collaborative filtering can be interpreted as a network of users and items, the answer to this question is yes. As a result, it should come as no surprise that graphs are a popular way to visualize collaborative filtering-based recommendation[3, 17], and the application developed for this thesis will also contain a graph-based visualization.

Herman et al. [20] define the graph drawing problem as follows: "given a set of nodes with a set of edges (relations), calculate the position of the nodes and the curve to be drawn for each edge". This thesis can then be seen as an effort to solve the graph drawing problem in a particular context, i.e., specific constraints in terms of screen size, data dimensionality, data quantity, et cetera.

1.3.5 Human-computer interaction and usability

In its most basic form, this is a thesis about humans and computers. The human uses a computer to find new item recommendations. The computer provides these recommendations. The human wants to understand the recommendation process. The computer explains how it computed the item suggestions. The user interacts with the system to further his/her understanding of the subject.

In this thesis we will try to understand the user, the system or computer, and the interface that allows interactions between them. Human interaction (CHI) is defined as "is the study of how people design, implement, and use interactive computer systems and how computers affect individuals, organizations and society."[54].

To improve the interaction between humans and computers, we will aim to develop an interface with high usability. The evaluation methods in this thesis draw from the evaluation techniques from the field of human-computer interaction. Paper prototyping, think aloud protocol, subjective evaluation methods among others, are examples of these techniques[39]. In this sense, this thesis can be seen as a case study of designing, implementing and incrementally improving an application using CHI techniques.

1.4 The visual explanation system

The application created for this thesis is a page action Chrome Extension that injects *HTML* and *JavaScript* into the recommendations page of *Last.fm* at `http://last.fm/home/recs`. The application makes use of several *JavaScript* libraries, such as D3[1] and jQuery[2], as well as a specific JavaScript library by Felix Bruns[3] to facilitate the usage of the Last.fm API[4].

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

1.5 Next chapters

The rest of this thesis text is organized as follows. First we will present a literature study on the topics discussed in section 1.3 of this introduction. The next chapter is a comparative study of recommender systems with visual explanation systems. After this we will try to design a white box for CF-based recommendation by applying some of the conclusions and techniques discussed in the literature study. This concludes the theoretical part of the thesis.

In the next part of the thesis we will look at the evaluation of the application's design through user tests, as well as the software design and implementation of the application. In these chapters we will try to identify liabilities in the design, highlight implementation details, and discuss evaluation methods.

The thesis text ends with an analysis of the application's evaluation results, further conclusions, and a reflection on future work and opportunities.

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

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

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

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

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

Chapter 2

Literature study

The term 'human-computer interaction' describes a phenomenon where two actors, the human or *user* and the computer or *system*, share a communication channel. The communication channel is a representation of data of the system towards the user. It is clear that both actors will impose certain restrictions on this visual communication channel[49, 61]. In this chapter we will try to establish who these actors are, what this communication channel looks like, and what kind of restrictions are imposed upon the visual communication channel.

2.1 The user

Who is the first actor? The answer to this question is of course very broad, so first we will try to list what we actually want to know. The goal is to allow the user to gain insight into the system through an interactive, visual explanation system. In conclusion, two specific questions arise:

- how does a human gain insight?
- what kind of limitations are imposed by the user on the design of an interactive visualization?

The following subsections try to establish an answer to these questions. Most of the ideas in these subsections are drawn from a papers by Yi. et al. [63], North et al. [42], Klein et al. [29, 30], and a book by Colin Ware [61].

2.1.1 Insight gaining

What is insight? In [42] 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, [42] and [63] list characteristics that allow a finer evaluation:

- **Complex**: insight is complex in the sense that involves large amounts of data 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;

- **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 quality of insight can then be determined by quantifying each of these characteristics[42]. The previously described collection of properties defines insight. Now we will look at the closely related concept of sensemaking. The next paragraphs describe how a user can arrive at insight.

Sensemaking

Sensemaking plays an important part in insight gaining [63]. The definitions for sensemaking may vary. We adapt the definition presented by [29] and [63].

In [29] 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"[29].

Based on the discussion in [30] and [63], 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 [29] 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.

Processes of insight gaining

Although sensemaking can play an important part in gaining insight, it is not the only path to arrive at insight [63]. Soo Yi et al. [63] identify four processes through which insight is established. Note that these processes are intertwined and often used together to generate insights. The processes are as follows[63]:

- **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;

- **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;
- **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[63].

Improving insight

Yi et al. [63] identify several ways in which the insight gaining process can be made more efficient. They list the system's interactivity, the quality of visual encodings and usability among others, as possible enablers for increased insight gaining. Naturally, improvident designs will act as barriers rather than enablers in the insight gaining process.

Interactivity of the system promotes the user's engagement into the dataset. Spending more time with the data will allow users to form more detailed and accurate hypotheses, and as a result greater insight[63]. At the same time, while using the visualization, the user will become moe 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[61].

Similarly visual encodings that are counter-intuitive will also increase the cognitive load. Other barriers on insight gaining are clutter, occlusion and data overload[63].

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[63]. 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"[58].

Note that usability should not be considered a one-dimensional property of a user interface. Nielsen identifies several characteristics of usability in applications[39]:

- **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.

2.1.2 Interactive visualization

As we now have a better understanding of what insight is, we will try to establish how the insight gaining process works through visual data mining and interactive visualization. The relation between insight gaining and data visualization has been pointed out in other research. Colin Ware [61] describes interactive visualization as an "internal interface between the user and the computer in a problem solving system". Keim [27] 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 [49], 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"[49]. In conclusion, the visual data exploration process can then be understood as a hypothesis generation process[27].

A reoccurring theme in visualization design is "overview first, zoom and filter, and details on demand"[27, 49, 61]. 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[27].

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[27, 61, 63].

In [61], 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

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[61].
- **Types of interaction**: different types of interaction will have a different influence on user performance.
- **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 factors has been evaluated and since, they are captured in various laws. We will discuss some of them, as listed by Ware et al. in [61].

The reaction time is given by the *Hick-Hyman law*: $Time_{reaction} = a + b \log_2(C)$ with C the number of choices and a and b empirically determined constants. This formula has been derived from experiments in which subjects had to press one of two buttons depending on the color of a light that was turned on or turned off[61].

The reaction-time may be influenced by many other factors such as the amount of visual noise, the distinctness of the signal and so on. If a person is allowed to make mistakes, the subject will respond faster, but at a cost of loss of accuracy[61]. When performing long and tedious search tasks, vigilance will become an important aspect as well[61].

In [61] different kinds of interactions are discussed such as selection, hover querries and path tracing. The time required for selecting an object in a two-dimensional space is determined by *Fitt's law*: $Time_{selection} = a + b \log_2(D/W + 1.0)$ with D the distance to the target, W the width of the target and a and b empirically determined constants[61].

A common way of selecting objects is through hover queries: the user drags a cursor over an object[61].

Another type of interaction is path tracing. The speed at which a user can trace or follow a given path is given by $v = W/\tau$ with velocity v, W the path width and τ a constant depending on the motor control system of the user[61].

When a task is repeated over time, the user will become more efficient at performing this task. Depending on the difficulty of the task, this learning effect will be more or less prominent. The *power law of practice* describes this speed up [61]: $\log(T_n) = C - \alpha \log(n)$ in which T_n is the time required to perform the task at the n-th trial, with $C = log(T_1)$ and α the steepness of the learning curve.

Some actions are easier to learn than others. This is related to the *stimulus-response (S-R) compatibility*. The stimulus-response compatibility refers to the way in which skills that have been learned before through everyday experience, are applied to computer control design. However it is not necessary to produce a whole virtual reality to create excellent computer interfaces. Research shows that mismatches between the interface and real world experience are not necessarily detrimental to the efficiency of human-computer interaction. Ware concludes that "it would be naive to conclude that computer interfaces should evolve toward VR simulations of real-world tasks (...). The magic of computers is that a single button click can often accomplish as much as a prolonged series of actions in the real world"[61].

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[61]:

- **Human**: the user gains understanding of 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 from data mapped onto the spatial model.

When exploring spatial maps, a user is confronted with the *focus-context problem*, i.e., "the problem of finding detail in a larger context"[61]. 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[61].

Ware and Mitchell remark that the human visual system is already well-adapted to the spatial focus-context problem[61]. Therefore, they argue that when designing a display, it should already try to take a maximal advantage of these perceptual skills.

The time it takes to navigate between two points in the data space to gather information is called the *cost of knowledge*[61]. To keep the cost of knowledge low, various interaction techniques have been developed, such as distortion, rapid zooming, elision, and multiple windows[61]. In section 2.3 we will take a closer look at some of these techniques.

Problem solving loop

The problem solving loop can be described through means of a *visual thinking algorithm*[61]. 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. Ware lists the following components[61]:

- **Early visual processing**: Early visual processing occurs at the lowest level, where elementary features such as color, texture information, local edges, and motion are extracted from the visual image.
- **Pattern perception**: Pattern perception occurs by combining the extracted features, patterns can be detected within the visual image.
- **Eye movements**: A map schedules tasks to explore proto-patterns through eye-movements. These proto-patterns correspond to the patterns that have a high probability to be relevant to the current task.
- **The intrasaccadic scanning loop**: Information is processed serially when focusing on an area of interest.
- **Working memory**: The working memory forms an intermediary between the incoming patterns and long-term memory. The currency of information in the working memory are *object files*. Object files are a combination of visual attributes and semantic meaning.
- **Mental imagery**: The ability to visualize internal images, whivh can be combined with external images, such as data visualization, to construct and test hypotheses about the visualized data.
- **Epistemic actions**: Epistemic actions are actions that support information search, such as eye movement, and mouse selections. Out of all epistemic actions, eye movements have the lowest cost.

- **Visual queries**: A visual query translates a hypothesis into a cognitive task. The result of this query can be a pattern or lack of a pattern. To reduce errors and increase efficiency, patterns are typically small, as the capacity of the visual working memory is rather limited.

- **Computational data mappings**: The computer maps data onto the screens. Sometimes the data is transformed before doing so. 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.

In [61], Ware and Mitchell list ten different visual thinking algorithms. We will describe our own visual thinking algorithm in chapter 3, based on these algorithms.

2.2 The system

Before describing the visual communication channel between human and computer, it is necessary to know what exactly has to be explained through visualization. Therefore we will take a closer look at the system, i.e., the recommender system.

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 [3, 5, 18, 37, 21].

One of the incentives behind creating recommender system is the 'long-tail phenomenon'. 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[45]. 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[45, 21]. Recommender systems then provide a means to find relevant items within this much larger range of items[45]. They enable to "connect supply and demand, introducing consumers to these new and newly available goods and driving demand down the tail "[1, 21].

The success of recommender systems to achieve this objective has been the subject of some research, e.g. [35] and [21]. Different classes of recommender algorithms favour different properties[5, 48]. This is elaborated further in subsection 2.2.3. For now its enough to see that the quality of recommendations is not uniquely defined. For example, in order to increase coverage of the item space, achieving high serendipity and novelty may be desired more than high recommendation accuracy, depending on the application context[48, 54, 21].

Typical applications of recommender systems are product recommenders for online

[1] `http://www.amazon.com/`

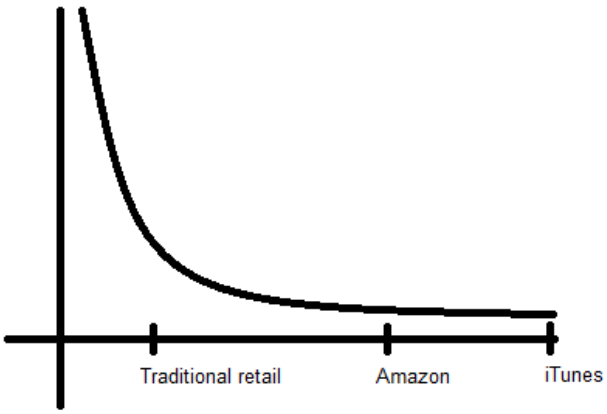


Figure 2.1: The long-tail: Items ordered by popularity are layed out against their popularity rating. Most of the items reside in the long tail of the graph. Companies such as Amazon can offer a vastly greater subset of the total item space.

retailers, movie and music recommenders such as *Netflix*[2] and *Last.fm*[3], and news article recommenders in online news services[35, 45, 21].

Recommender system have opened up new possibilities in the landscape of online retail, and as a result, spurred the interest of businesses in this field. A remarkable initiative was the Netflix challenge. In 2006, Netflix Inc. offered a prize to beat the performance of their recommendation algorithm by 10 percent. It gave a significant boost to the research on recommendation algorithms, and yielded a winning algorithm in September 2009[2, 45].

2.2.1 Properties of recommendation algorithms

in [19] and [48], Herlocker et al. and Shani et al. respectively, compare the performance of recommender systems. They list a number of metrics for recommendation algorithms. The resulting set of properties consist out of the following characteristics:

- **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.
- **User preference**: The opinion of certain users may be more valuable than the opinion of others.
- **Coverage**: The proportion of items that the recommender system can recommend is referred to as catalog coverage. Another measure in this respect is the percentage

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

of all items that are recommended to users. Finally we can also look at the diversity of the recommended items. Coverage can also mean the proportion of users or user interactions for which the system can recommend items. The cold start problem relates to coverage as it measures the coverage for a specific type of users, namely new users.

- **Confidence**: Confidence in the recommendation can be defined as the systems trust in its recommendations or predictions. The most common measurement of confidence is the probability that the predicted value is indeed true, or the interval around the predicted value where predefined portion of the true values lie. Confidence bounds can be used to filter recommended items where the confidence in the predicted value is below some threshold.
- **Trust**: Trust refers to the user's trust in the system, as opposed to confidence.
- **Novelty**: Novel recommendations are recommendations for items that the user did not know about.
- **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.
- **Diversity**: Diversity is generally defined as the opposite of similarity. Note that an increase in diversity may correlate to a decrease in accuracy.

2.2.2 A classification of recommendation algorithms

Based on classifications presented in [5] and [21], 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)[18, 45], 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[5, 3].

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[5, 18]. A typical user profile in a collaborative system consists of a vector of items and their ratings, that continuously augmented as the user interacts with the system over time[5].

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[45]. 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[45].

In order to calculate the blanks, there is a variety of similarity functions that has been developed over the years. Often some sort of distance function is used to compute

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

Figure 2.2: The utility matrix A.

distance between profile vectors. If the distance is small, profiles will most likely have a high similarity[45]. 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[5, 44].

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 some kind of stemming system[45]. *Stemming* is a technique to use root forms that capture a common meaning of certain words, rather than words themselves. For example 'compute', 'computation', 'computer', and 'computes' all have a reference to an underlying concept. Words are given a quantification of relevance to this root form, the so-called TF.IDF (term frequency times inverse document frequency) score. This score is computed in terms of the frequency of the word in a document, number of documents in which it occurs, and the total number of documents. The resulting structure consists out of tuples of root forms and quantification values[44, 45]. The words with the highest scores are the words that characterize the document[45]. A downside of stemming is that the process may cause the loss of contextual information for each word[44].

In [3] and [37] web crawlers are used to gather and extract features from online documents. In [37] the web crawler collects properties such as movie title, director, cast, genre, plot summary, plot keywords user comments, reviews and so on, of each movie in the data set. Each property or feature is a 'bag of words' that is used in a naive Bayesian text classifier. This way each item can be categorized and the profile can be 'learned'[37].

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[45]. 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[10].

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

Mathematical models of music and images also allow for feature extraction. For example, in a study by Johnson et al. [24], *wavelets* were used to construct feature vectors from high resolution paintings. These were in turn used to classify paintings based on several clasification algorithms. A training set of Van Gogh paintings was used with the objective to classify other paitings as either genuine Van Goghs or forgeries. The algorithms were able to actually classify paintings with a high accuracy[24].

Similarly algorithms have been developed to classify music based on content features. There are various types of acoustic features that can be extracted. In [36] a distinction is made between rhythmic content features, pitch content features and timbral content features.

The research performed by Tzanetakis et al. in [57] and [56] is based on a the MARSYAS system. This system is an application that uses various techniques, such as Mel-Frequency Cepstral Coefficients (MFCC) and Short-term Fourier Transform Features (FFT) to retrieve acoustic features from audio signals [55]. In addition to these features, also wavelet-based feature extraction is used to retrieve rhythmic features from audio content [57, 56]. Using this system, music files could be classified into different genres based on rhythmic features[57].

A similar research is conducted by Li et al. in [36]. In addition to previously mentioned methods, also lyric-based classification is used. Apart from music classification, also music similarity search is performed. In this case, the feature vectors from the classification using MARSYAS features and wavelet-histograms can be used to form a hierarchy in which similar audio signals can be detected [36].

Hybrid recommendation

Hybrid filtering combines two or more recommendation algorithms[5]. In [5] a number of hybrid recommendation strategies are discussed. Burke et al. list seven different approaches for combining recommendation algorithms:

- **Weighed**: The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
- **Switching**: The system switches between recommendation techniques depending on the current situation.
- **Mixed**: Recommendations from several different recommenders are presented at the same time.
- **Feature combination**: Features from different recommendation data sources are thrown together into a single recommendation algorithm.
- **Cascade**: One recommender refines the recommendations given by another.
- **Feature augmentation**: Output from one technique is used as an input feature to another.
- **Meta-level**: The model learned by one recommender is used as input to another.

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 [5].

2.2.3 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.

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 [5]:

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

Both of these issues translate themselves into a sparse regions in the utility matrix. It is worth noting that content-based recommendation algorithms suffer less from the *new item* problem, as these tend to rely on features that are inherent to the items themselves, rather than user generated content. This is one of the reasons hybrid approaches can provide a solution to collaborative filtering[5]. For example, in [37], content-based predictors are used to create pseudo-user ratings to reduce sparsity of the utility matrix, used in a collaborative algorithm.

A problem that is typical of collaborative filtering is the 'gray sheep problem'[5, 18]. 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[5].

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[64]. This problem decreases the acceptance by the user of item suggestions. One of the solutions for this problem, proposed by Herlocker et al. in [18], 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[18].

2.3 The interface

In this section we will take a closer look at the visual communication channel.

Shirley et al.[49] lists three distinctive limitations:

- **Computational capacity**: time complexity and memory usage of algorithms must allow a responsive user interface, especially in the case of interactive visualization.

This is also a requirement to meet the *principle of transparency* discussed in section 2.1.2.

- **Display capacity**: there is a trade-off between the benefits of maximizing the *information density*, i.e., the measure of the amount of encoded information against the amount of unused space, and causing visual overload.
- **Human perceptual and cognitive capacity**: optimizing the cognitive cost is one of the key aspects that make up a successful visualization, as visual and non-visual memory capacity are limited[61], as we saw earlier in section 2.1.2.

There has been done extensive research in this domain. Over the last two decades various new information visualization techniques have been developed. These techniques can be classified based on three criteria [27]:

- **Data**: a classification based on the structure and type of data;
- **Technique**: a classification based on characteristics of visualization techniques;
- **Interaction and distortion**: a classification based on the way in which interaction between user and visualization is enabled.

2.3.1 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[27].

There are different types of data 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[27]. There are three different kinds of dimensions, namely[49]:

- **Quantitative**: numerical data on which arithmetic can be applied;
- **Ordered**: an enumeration that has a definite order;
- **Categorical**: data that has no specific ordering, and is distinguished by name only.

Relational data on the other hand consists out of nodes and links or 'edges'[27, 49]. Both nodes and edges can have associated attributes.

In [27] also text and hypertext, and algorithms and software are discussed as examples of other types of data. In the case of text and hypertext, standard visualizations are hard to use, as they cannot be described easily in terms of numbers. As a result, the data is first transformed into description vectors. Next, these vectors can be used in a visualization. Examples of software and algorithm visualization are flow diagrams, presentation using a graph-based structure of source code, and so on[27].

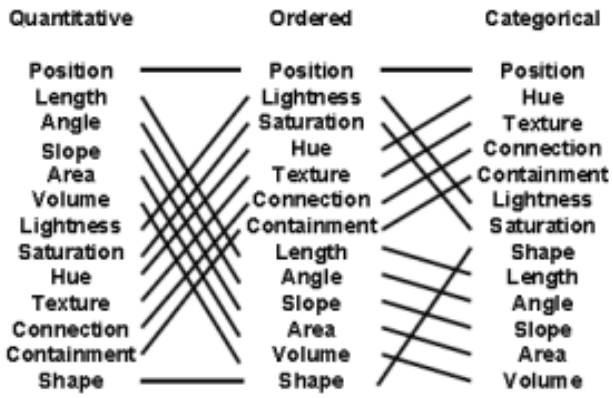


Figure 2.3: Visual encoding performance for each data type, ordered from best to worst.

2.3.2 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[49].

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[49]:

- **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 color, 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 [49]. Figure 2.3 gives an overview of the performance for each category, adapted from [49]. Note that spatial position is the most accurate for each data type.

In [49] 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:

- **Quantitative data**: uses a color map, a range of color values that can be continuous or discrete. It is recommended to use lightness instead of hue, as lightness has an implicit perceptual ordering. Moreover the human eye responds most to strong luminance. Hue on the other hand has a small range (around twelve values that can be reliably distinguished, including background and neutral colors);

- **Ordered data**: lightness and saturation are advised. As mentioned before, these have an implicit perceptual ordering;
- **Categorical data**: hue can be successfully applied for categorical data, keeping in mind its small range. An important remark is that roughly 10% of men is red-green color deficient. If a coding uses red and green, it may be wise to apply redundant coding using lightness or saturation in addition to hue [49].

Spatial layouts form other visual channels. Although these tend to be the most accurate, spatial layouts in two and three dimensions have several weaknesses [49]:

- **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;
- **Perspective distortion**: again, in the case of the mapping of abstract dimensions onto spatial positions, distances may convey meaning that may be distorted through perspective.
- **Text in arbitrary orientations**: special care has to be taken with text, as it may become very hard to read depending on the orientation.

2.3.3 Techniques

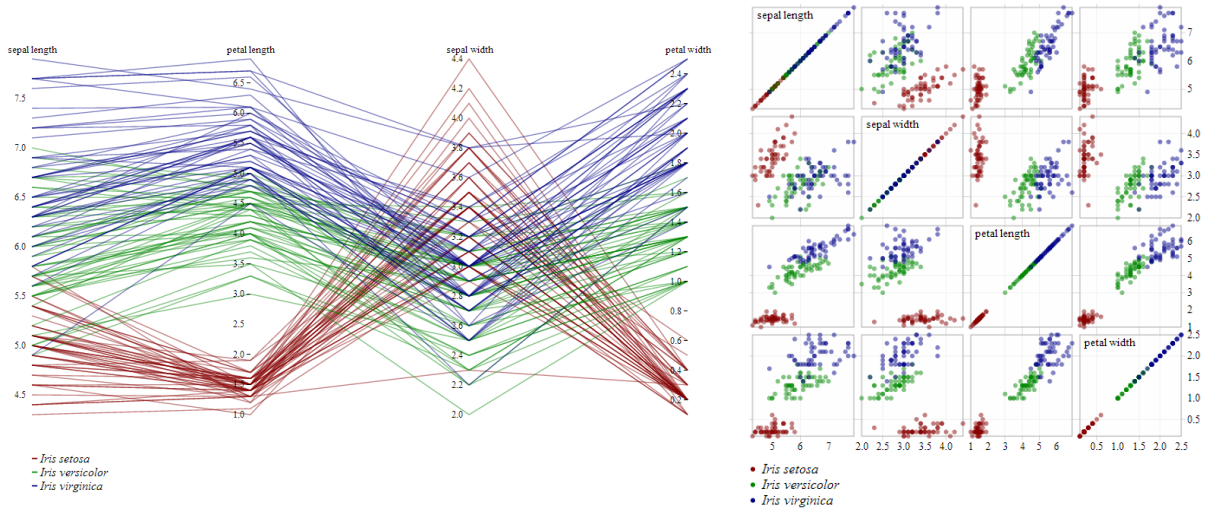
Visualization techniques try to deal with the limitations listed in the introduction of this section. Tamara Munzner [49] acknowledges that there is a trade-off to be made concerning the interaction cost: on the positive side, interaction allows for data exploration in a larger information space than could be understood in a single static image. The downside is that interaction costs, other than the obvious implementation and computational costs, include the requirements for human time and attention[49]. Since attention drops significantly after the first hour[61], it might become a critical factor in the quality of the visualization. This view is also supported by Ware's findings described in section 2.1.2, as eye-movements have a much lower cost than other epistemic actions[61, 62].

Based on the discussion by Ware and Mitchell in [61], Keim in [27], and Munzner in [49], we will discuss a number of visualization, interaction, and data reduction techniques.

Visualization techniques

In [27], Keim lists a number of visualization techniques:

- **Standard 2D and 3D techniques**: one-to-one data mappings from the data space onto the screen. For example xy- and xyz-plots, bar charts, pie charts, donut charts, line graphs and so on;
- **Geometrically transformed displays**: transformations of multidimensional spaces onto a two-dimensional display. For example *parallel coordinate technique*, as shown in figure 2.4a, or *scatterplot matrices*, as shown in figure 2.4b.



(a) Parallel coordinates.

(b) Scatterplot matrix.

Figure 2.4: Parallel coordinates and scatterplot matrix visualizations for Edgar Anderson's *Iris* data set. Taken from `http://mbostock.github.io/d3/talk/20111116/iris-parallel.html` and `http://mbostock.github.io/d3/talk/20111116/iris-splom.html` respectively.

- **Iconic displays/glyphs**: representations of a multidimensional set of attributes in a single icon[27, 61].
- **Dense pixel displays**: representations of a multidimensional data space where each dimension corresponds to a certain colour, and each dimension value is mapped onto a pixel on the screen. This way, large amounts of data can be visualized on a single display.
- **Stacked displays**: displaying multidimensional data by creating a hierarchy of stacked coordinate systems. For example using one coordinate system as the frame of reference for objects. Within these objects, smaller coordinate systems can be constructed, displaying more detailed information for the other dimensions of the data. This can be applied recursively[27].

Interaction techniques

The previously described vsiualization techniques can of course be combined with various interaction techniques[27]. When applying visual thinking algorithms on a visualization, the focus-context problem, described in section 2.1.2, will surface.

Several techniques can be applied to solve the focus-context problem further; for example distortion and rapid zooming[61]. Keim [27] also lists a number of techniques for interactive visualization. A summary based on [61], [49] and [27] is presented here:

- **Dynamic projections**: Data projections are dynamically changed, allowing the user to explore a multidimensional data set.

- **Interactive filtering**: The data set is interactively partitioned into different segments in order to focus on different subsets through querying, browsing or more advanced methods.
- **Interactive zooming**: The data representation is interactively changed, moving between different resolutions. A distinction can be made between two kinds of zooming[49, 61]:
  - **Geometric/rapid zooming**: Rapidly changing viewpoint in a two- or three-dimensional space, similar to adjusting the ratio of the focal length on binoculars;
  - **Semantic zooming/elision**: Changing the representation of an abstract object visualization, for example expanding and collapsing parts of a large structure that was represented by a single visual entity.
- **Interactive distortion**: Parts of the data are displayed with a high level of detail (focus), while others are shown with a lower level of detail (context). Examples are 'fish-eye' distortion, bifocal displays and hyperbox.

Data reduction techniques

In [49] four main strategies are distinguished to reduce the amount of data shown, avoiding visual clutter:

- **Overviews and aggregation**: many data sets have an internal structure at different scales, allowing multi-scale representations. With aggregate representations many items can covered by a single mark;
- **Filtering and navigation**: by filtering only a subset of the data is shown. Navigation is a special type of filtering based on spatial positioning. In geometric zooming the viewpoint can be changed in the two- or three-dimensional space. In semantic zooming through a space of abstract items the object visualization may change very much depending on the resolution, i.e. the number of pixels available;
- **Focus and context**: changing the data representation on different resolutions. E.g. fish-eye technique;
- **Dimensionality reduction**: may be effective if there are many dimensions in the data. In [49] two techniques in particular are described:
  - **Slicing**: a single value is chosen from the dimensions to eliminate, and only the items matching that value for the dimension are extracted to include in the lower-dimensional 'slice'.
  - **Projection**: no information of about the eliminated dimensions is retained; the values for those dimensions are simply dropped.

2.3.4 Graph-based visualization

In this subsection we will take a closer look at *graph-based visualization*. As discussed in section 2.3.1, relational data is data that has an inherent relation among its elements[49]. "Given a set of nodes and a set of edges, what are the positions for each node and the curves for each edge?"; this is basically the essence of the graph drawing problem[20].

Scalability is one of the central issues with graph drawing, as graph size poses several important challenges[20]:

- **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.

To determine the best way to draw a graph, a definition of properties and a classification of layouts is required. These properties will impose constraints on the algorithms used for the layouts presented. For example planarity of the graph, i.e. the absence of edge-crossings, predictability, density, and aesthetic constraints (e.g. symmetry) can be imposed on the final layout. Note that some of the 'aesthetic rules' can have a major impact on the usability of the graph as well; in that case they are said to have an absolute character[20].

Traditional layouts

In [20] a number of traditional graph layouts are discussed. Here we give a brief overview of some of them:

- **Tree layout**, **H-Tree layout**, **radial layout**, **balloon layout**: these layouts are typically used for drawing trees. The algorithms to construct these layouts are of low-complexity, i.e. $O(N)$ with N the number of nodes;
- **Sugiyama layout**: this is a layout designed specifically for directed graphs. The complexity of the algorithms for this layout depends mainly on the edge-crossing minimization. As this an NP-hard problem, heuristics are usually applied.
- **Spring layout (force-directed methods)**: this layout is applied to draw undirected graphs. Nodes and edges of the graph are modeled as physical bodies tied with springs. The time complexity of the algorithm and quality of the result may vary depending on the physical models used. Still, the best variants are estimated to work with a complexity of $O(N^3)$ with N the number of nodes, which is relatively slow. Another disadvantage is that the layout is rather unpredictable, which may be less preferable for information visualization. An example of a force-directed graph can be seen in figure 2.5.

Figure 2.5: A force-directed graph layout. Taken from `http://bl.ocks.org/mbostock/4062045`.

Three dimensional layouts

Limits to information visualization with graphs depend largely on the size of the graph, as we have seen in the introduction of this section. To increase graph size, one can work with some kind of interaction technique[20, 62]. However, as discussed in section 2.1.2, it is better to take advantage of the efficiency of eye-movements by maximizing the amount of displayed data[61, 62]. Ware et al. conclude that the question how large a graph can be seen in a non-interactive display is an important one[62].

Three dimensional graph visualizations allow for larger network structures to be seen, as the extra dimension will provide more 'space'[20, 62]. Distance can be perceived as the result of a series of depth cues, such as stereoscopic disparities, kinetic depth, perspective, texture and size gradients, occlusion, shape from shading, among others. From these depths cues, Ware et al. [62] identify the user's stereoscopic disparities and kinetic depth cues as the most significant factors that determine the performance of three dimensional graph visualization. Disparities are the relative differences between pairs of features imaged in the two eyes, and form the basis for stereoscopic depth. They allow for extremely fine judgments. The kinetic depth effect is caused by rotating a three dimensional wire-object: it will appear strongly three dimensional, even if the rotation has been stopped. Overall, stereoscopic depth appears to be the dominant factor [62].

In [20] a number of three dimensional layouts are listed:

- **Generalize two dimensional layouts (e.g. radial layout)**: most of the previously described traditional methods can be transformed to three dimensional counterparts;
- **Cone tree**: A node is placed at the apex of a cone with its children placed evenly along its base. In the original implementation, each layer has cones of the same height and the cone base diameters for each level are reduced in a progression so

that the bottom layer fits into the width of what the authors called the room, i.e. the box containing the full cone tree;

- **Real-world metaphors**: Because of general human familiarity with 3D in the physical world, 3D lends itself to the creation of real-world metaphors that should help in perceiving complex structures.

Data and dimensionality reduction techniques

As graph size is one of the biggest issues of graph visualization, techniques have been developed to apply data reduction techniques on graphs[20, 49]. Apart from the traditional distortion and data transformation techniques, there are some techniques that are specific for graph-based visualization. Based on a survey by Herman et al. [20], and papers by Holten [22], and Holten et al. [23], we give an overview of two techniques: clustering and edge-bundling.

Clustering is the process of discovering groupings or classes in data based on chosen semantics. Clustering techniques can be divided into two categories:

- **Structure-based clustering**: clustering based on structural information, for example choosing clusters that have the least number of edges between them. Structure-based clustering tends to retain the structure of the original graph;
- **Content-based clustering**: the use of semantic data associated with graph elements. This is usually harder as this data is application-specific.

Node metrics, i.e. a metric associated with a node in a graph, may be applied to cluster a graph. One can distinguish between structure-based and content-based metrics[20].

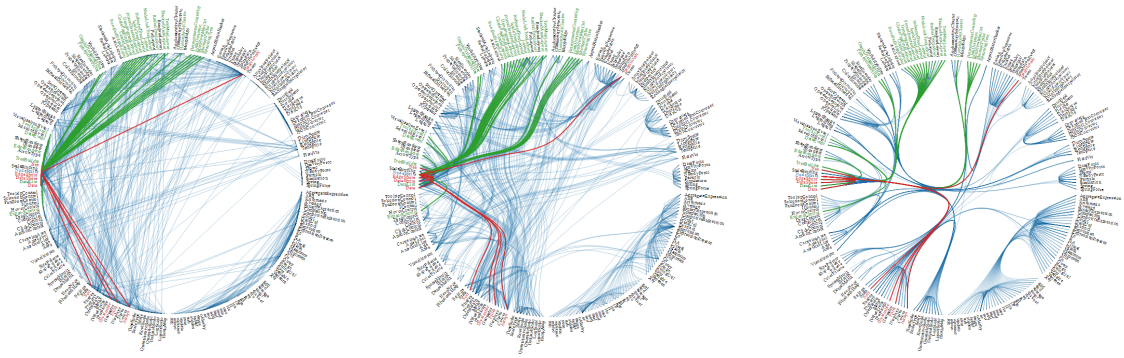
Compound graphs can be used to bundle edges and nodes into 'super-nodes'. This graph can then be navigated, instead of the original one. In structure-based approaches, the compound graph provides a good overview of the global structure of the original graph[20].

Hierarchical clustering is achieved by successively applying the same clustering process to groups discovered in the precious step of the clustering process. This way a hierarchical structure may be induced upon a graph that would otherwise not necessarily have one[20].

Once a subset of nodes has been selected, a method of representing the unselected nodes must be chosen. In the case of clustering, the selected set of nodes is the set of super-nodes or the groups themselves. Kimelman et al. name three possible approaches[20]:

- **Ghosting**: deemphasizing nodes, or relegating nodes to the background.
- **Hiding**: simply not displaying the unselected nodes. This is also referred to as folding or eliding.
- **Grouping**: grouping nodes under a new super-node representation.

In [22] Danny Holten describes an edge-bundling method to visualize compound graphs. He describes the process as follows. The hierarchy is shown via a standard tree visualization method. Next, each adjacency edge is bent, and modeled as a B-spline curve, toward the polyline defined by the path via the inclusion edges from one node to another.



(a) Low edge bundling strength. (b) Medium edge bundling strength. (c) High egde bundling strength.

Figure 2.6: 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.

This hierarchical bundling reduces visual clutter and also visualizes implicit adjacency edges between parent nodes that are the result of explicit adjacency edges between their respective child nodes.

Edges are modelled as "flexible springs that can attract each other, similar to how electrical wires are bundled within a network"[23]. Figure 2.6 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.

In another paper by Danny Holten and Jarke van Wijk edge-bundles for more general graphs, i.e. not necessarily hierarchical graphs, are discussed. It uses a self-organizing approach in which edges are modeled as flexible springs that can attract each other. Their visualization technique achieves very good clutter reduction and allows to discern high-level edge patterns [3].

2.4 Related work

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

Next a comparison is made between these systems, based on a number of objectives for explanation systems listed by Tintarev and Masthoff in [53].

2.4.1 PeerChooser

PeerChooser is a "collaborative movie recommender system with an interactive graphical explanation interface"[43]. It aims to address the black box problem, described in section 2.2.3[43].

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

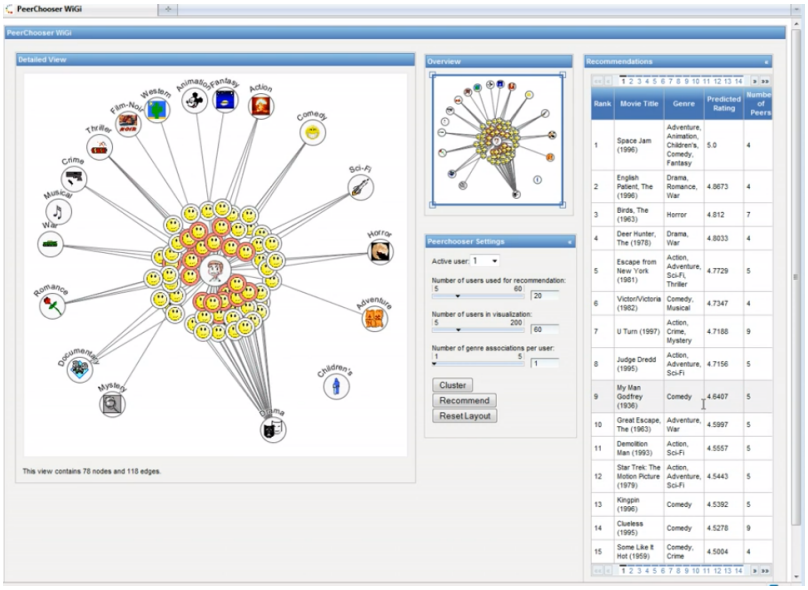


Figure 2.7: The PeerChooser interface.

of the data, extra cluster nodes are added to the vsiualization. These cluster peer nodes by genre[43].

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[43].

Figure 2.7 shows the graph-based interface of the PeerChooser application.

2.4.2 Pharos

Content-centric social websites, such as blogs and discussion forums, contain vast amounts of fast growing information. Recommendation systems have been developed to help users find the information they are looking for. The *Pharos* application tries to address two distinctive problems that present themselves in this context: the cold start problem and the black box problem[64], as described earlier in section 2.2.3.

As they hope to overcome previously defined problems, Zhao et al. [64] collect and visualize content-related social behaviour. The resulting data set is tranformed into a social map. The social map provides a context for new users, addressing the cold start problem. Secondly, the user can explore the social map to increase understanding and user interaction, in an effort to overcome the black box problem.

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

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

Figure 2.8: The Pharos social map. Colours indicate activity within a certain group.

An example of what the resulting visualization looks like, is depicted in figure 2.8.

2.4.3 SFVis

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

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

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

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

2.4.4 Smallworlds

In [17], 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[17].

SmallWorlds is "a visual interactive graph-based interface that allows users to specify, refine and build item-preference profiles"[17]. The system promotes transparancy in the recommendation process, and gives the user a sense of control over the recommendation

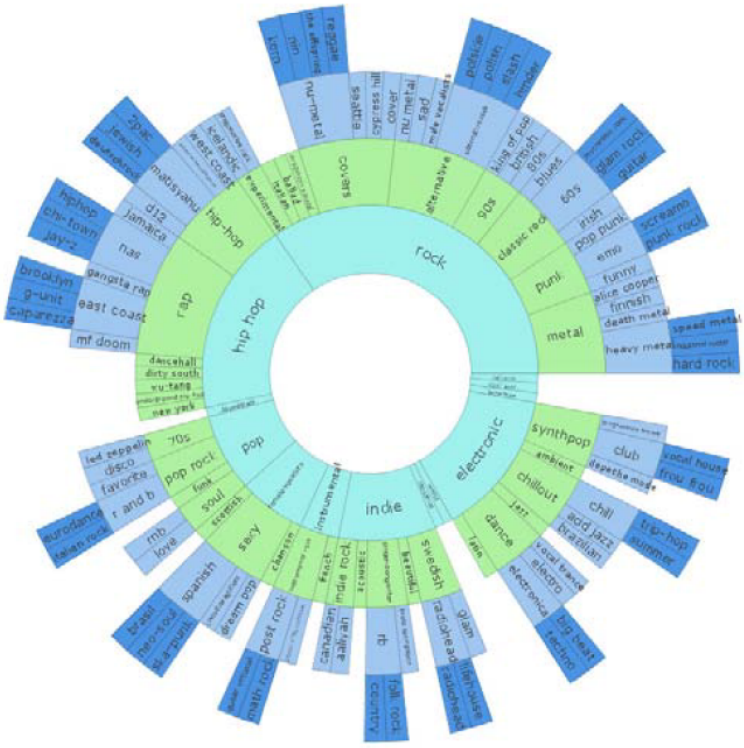


Figure 2.9: SFViz graphical user interface: tag tree.

process through interactions. This way, Gretarsson et al. try to further overcome the limitations of their recommender system[17].

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.

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 2.10 shows a screenshot of the application.

2.4.5 TasteWeights

TasteWeights is a hybrid recommender with an interactive graphical user interface[3]. The application allows the user to express his/her preferences changing the weights of incoming data sources.

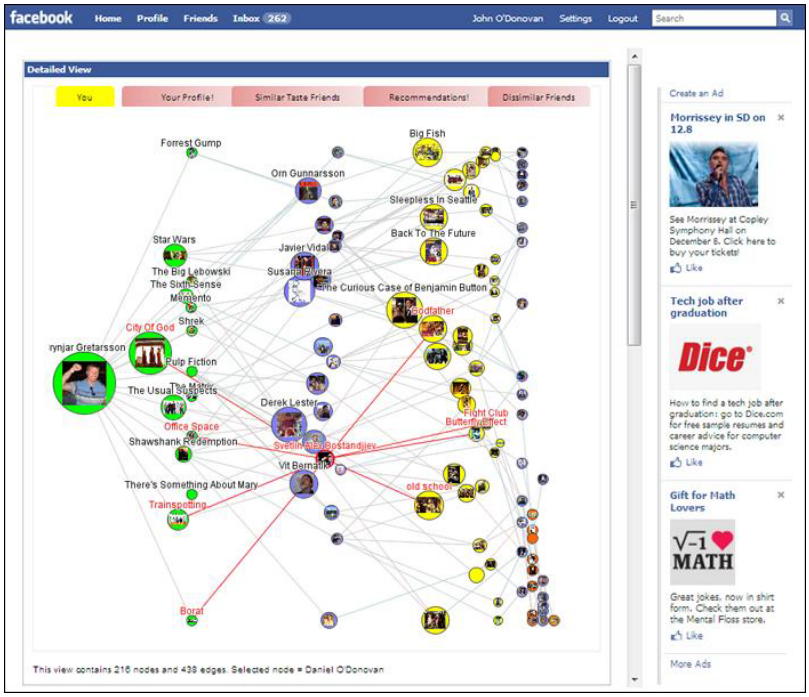


Figure 2.10: The SmallWords interface.

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"[3] (cf. Smallworlds). They introduce two enhancements for the traditional techniques. Multiple web sources, namely from *Facebook*[5], *Twitter*[6] and *Wikipedia*[7] are combined when computing the recommendation. This combination provides a hybrid of different recommendation strategies, namely: collaborative filtering, expert-based and content-based respectively. The second enhancement is a new user interface that provides transparency into the recommendation process.

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

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

Figure 2.11 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[3].

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

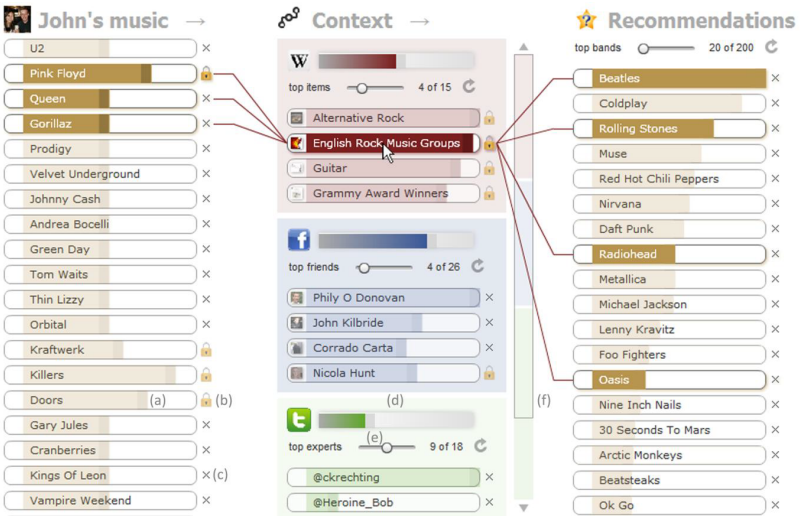


Figure 2.11: The TasteWeights interface.

Table 2.1: A comparison of the recommender systems, based on recommender system properties.

2.4.6 Comparative study of visual explanation systems for item recommendation

The following listing is adapted from [53]:

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

In table 2.1 an overview is presented of some the different recommender systems and some of their properties. So far most of the applications discussed here used collaborative filtering and some kind of visualization and allowed the user to interact with the system.

Table 2.2 gives an overview of each of the techniques that were used to design and implement the visualizations.

Table 2.3 gives an overview of the performance of each visual explanation system based on the criteria listed by Tintarev and Masthoff in [53].

Table 2.2: A comparison of the visual explanation systems, based on visualization, interaction, and data reduction techniques.

Table 2.3: A comparison of the visual explanation systems, based on the criteria by Tintarev and Masthoff listed in [53].

Chapter 3

Designing a white box model for collaborative filtering

In this chapter we will describe our approach to design a white box model for collaborative filtering. In order to provide explanations about the recommendation process, we will base our design on the characteristics of collaborative filtering.

3.1 The visualization

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 $U \cap I = \emptyset \wedge E \subseteq U \times I$[7]. Each non-blank entry in the utility matrix will then correspond to an edge. Figure 3.1 shows how a matrix is transformed into a dual graph.

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

One of the challenges of this approach is overcoming the graph drawing problem, as defined earlier in section 2.3.4. In an application where easily thousands of items may be involved, scalability becomes a significant constraint on the visualization design[20].

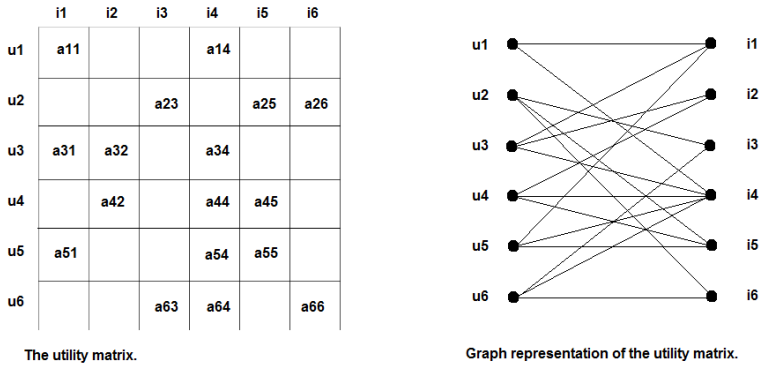


Figure 3.1: Transforming the utility matrix into a dual graph: two distinct sets of nodes, users and items, only share edges between nodes of different sets.

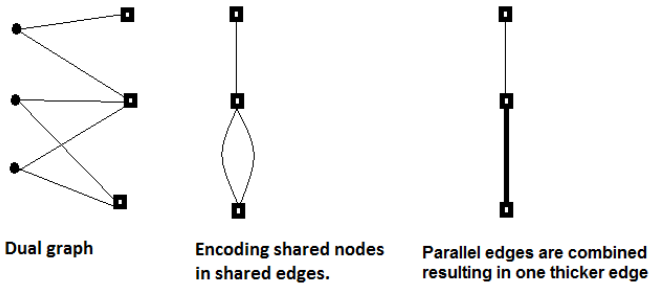


Figure 3.2: A row reduction operation on each pair of edges in a dual graph will result in a dimensionality reduction where one set of nodes is removed from the graph. An additional data reduction can be achieved by clustering edges into a thicker edge. Edge thickness then depends on the number of edges involved.

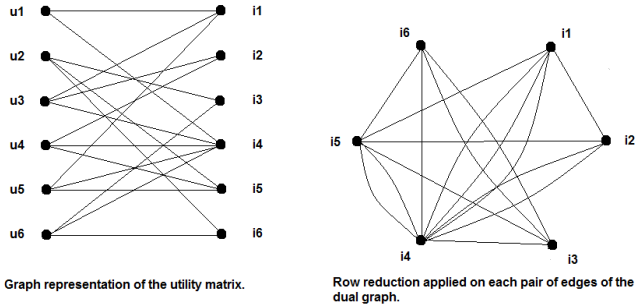


Figure 3.3: Row reduction applied on the graph in figure 3.1.

Several strategies have been identified to reduce the number of items, reduce the number of dimensions and reduce visual clutter, as listed in section 2.3.3 and 2.3.4.

Based on a visualization design by Valdis Krebs in [52], 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 3.2 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[31, 52].

For this white box model we will not apply the final step of Krebs' graph design. Instead we will keep parallel edges 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 3.3 shows how the dual graph from figure 3.1 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. 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

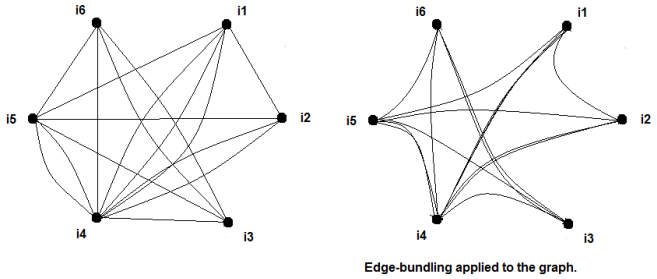


Figure 3.4: Edge bundling applied on the graph in figure 3.3.

graph.

Next we will apply edge-bundling on the graph to improve discernability and reduce visual clutter[22, 23]. Figure 3.4 shows how this applied on the graph from figure 3.3.

From a graph drawing perspective we managed to reduce visual clutter by reducing the number of displayed items and applying edge-bundling[20, 22]. Of course cognitive aspects come into play as the visualization encodes now more implicit information that needs to be interpreted by the end user[20, 61].

In order gain insight into the recommendation process, it is important that certain contextual information does not get lost through dimensionality and data reduction techniques. 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.

To accomplish this, the active user's neighbours should be included in the visualization in one way or the other. One solution would be to label each of the edges with the corresponding neighbour's username. However, neighbour names may overlap each other, or occlude parts of the graph. Instead, another idea would be similar to the PeerChooser application's layout. Re-introducing neighbours into the graph on demand. The corresponding edges could then be restored temporary. By adding and removing other users, item profiles can be compared.

To keep things simple, the user's top neighbours are listed next to the graph in the current visualization design. 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. In essense it shows the same information on demand as in the previously described alternative, but in a more implicit way.

3.2 The visual thinking algorithm

We will try to approximate a visual thinking algorithm applied by users when exploring the graph presented in section 3.1. The algorithm in table 3.1 is the degree-of-relevance highlighting algorithm derived by Ware and Mitchell [61].

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 (infromation scent).*
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.
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 3.1: Degree-of-relevance highlighting visual thinking algorithm by Ware and Mitchell [61].

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 2.1.1, to get a better understanding of the user.

3.3 User profile

3.4 Use cases

Chapter 4

Iterative development

This chapter describes how the idea presented in chapter 3 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"[51]. It is a technique for designing, testing, and refining user interfaces [50], and is closely related to usability testing [51]. In the last decade it has become a regularly applied technique in major businesses such as IBM, Digital, Honeywell, and Microsoft among others[50].

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[8].

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[8].

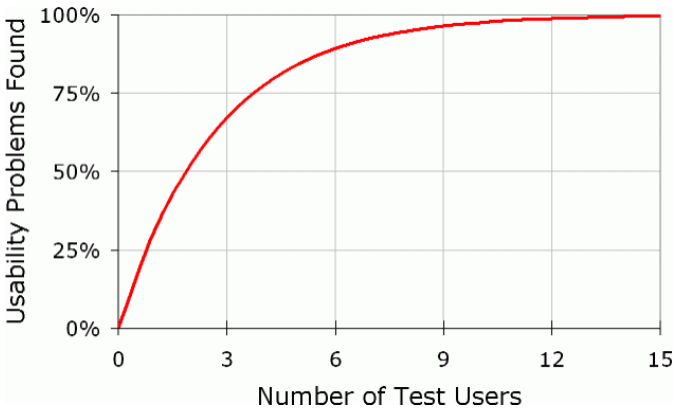


Figure 4.1: The curve shows the user test's diminishing returns beyond a certain amount of test users; adapted from `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`.

The methods we will be using here are a variation on *usability engineering* called *think aloud* user tests, and a special type of questionnaires called *system usability scale* (SUS) *questionnaires*.

In order to perform reliable usability tests, the test users have to be representative for the actual user population[8]. 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[50].

The number of users can often be limited to a certain amount. As the number of detectable problems is likely to be finite, from a certain point on adding more users to the usability test will not produce new or better results[8, 41]. The graph in figure 4.1, adapted from [41], 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[41].

Usability engineering

In [8], 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[8].

Using usability labs can be rather costly, as 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'[8]. During a think aloud test, the user describes his/her reasoning for each action he/she undertakes[40]. This method has several advantages and disadvantages, as listed in table 4.1, based on

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.

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.

[40] and [50].

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 [28].

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[47]. The ten questions are listed in appendix A.

4.2 Story board

4.3 Iteration 1: paper prototype

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

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

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

Chapter 5

Implementation: the SoundSuggest application

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

5.1 Technologies

5.1.1 Chrome extensions

Chrome Extensions are applications written in *HTML*, *JavaScript* and *CSS*, that enhance to functionality of the Google Chrome web browser[14].

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[12]. 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[15].

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[13].

Many extensions use a two-layered structure in which you have a background page and UI pages or content scripts[15]. 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[11]. 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[15].

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[15].

[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[15].

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[34].

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[33].

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*[38, 59].

D3.js is a JavaScript library that uses the W3C standards *HTML*, *SVG* and *CSS* to build data-driven documents[4]. 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"[25].

- **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"[26].
- **Purl.js**[6]: a library built on the jQuery library to retrieve GET parameters from the web page's URL.
- **Spinner.js**[7]: a library that creates a spinner element with given parameters for customization.

5.2 Software design and application architecture

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

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

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

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

[5]

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

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

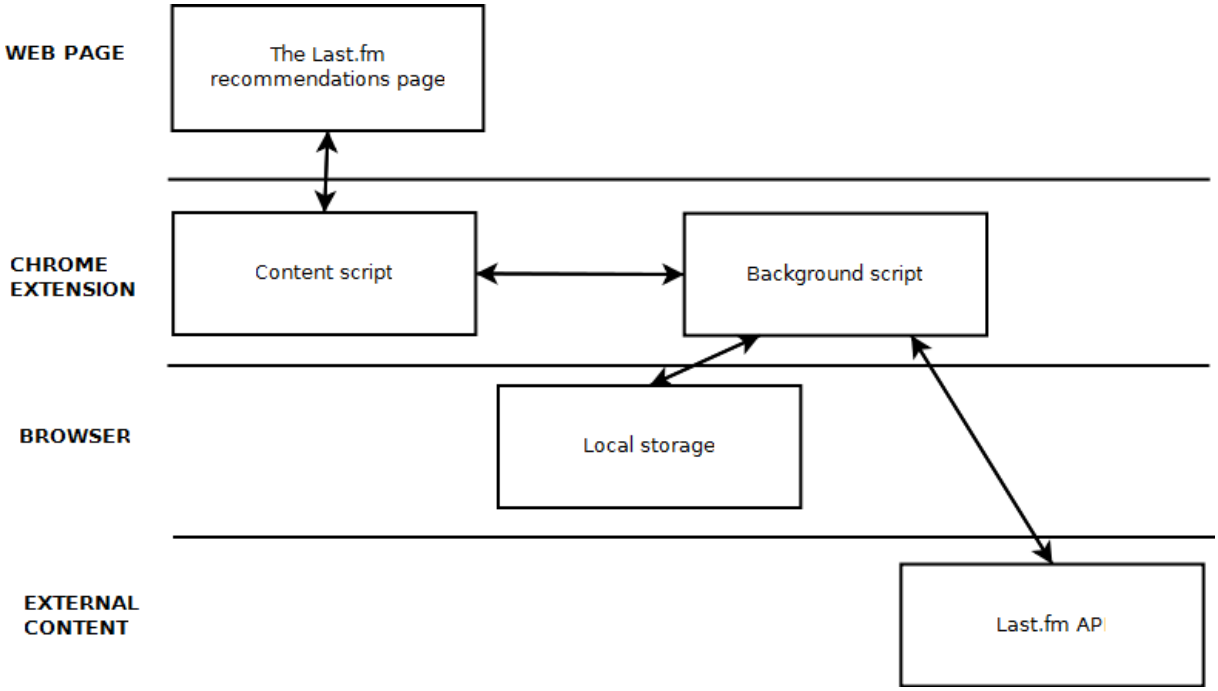


Figure 5.1: The architecture of the application.

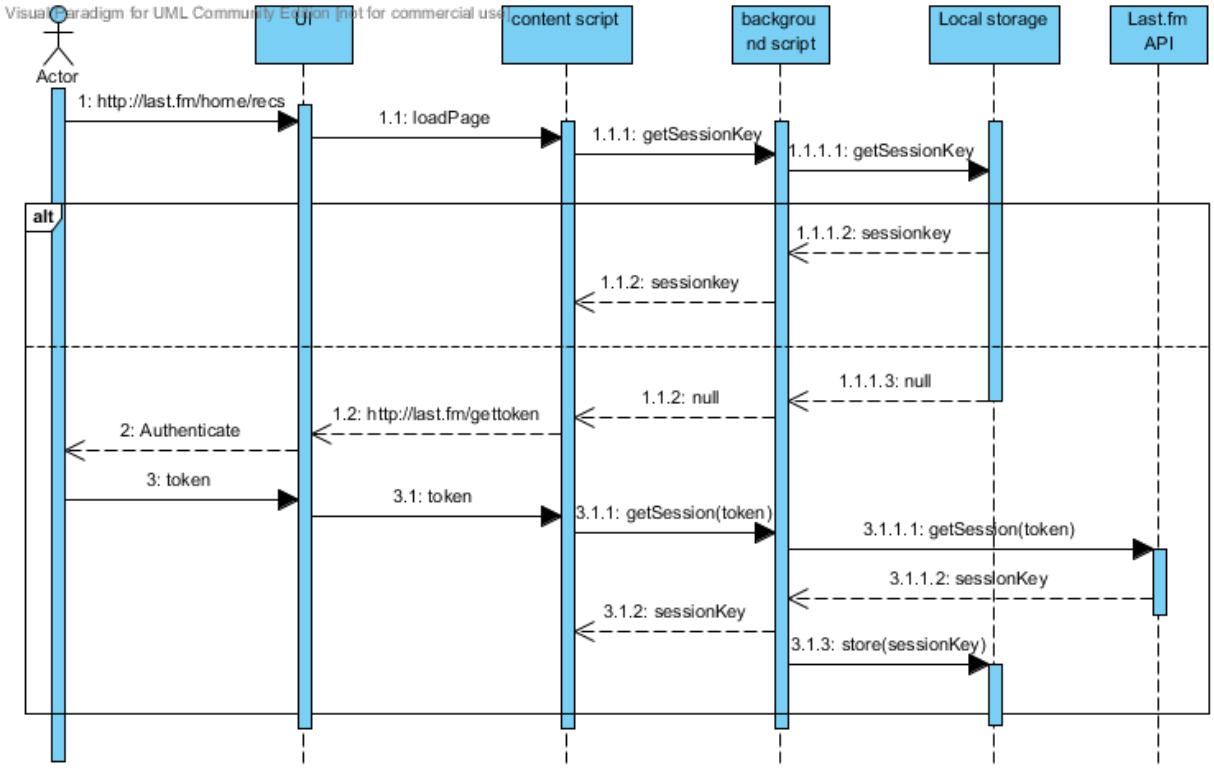


Figure 5.2: Sequence diagram: opening the Last.fm recommendations page part 1: retrieving a session key.

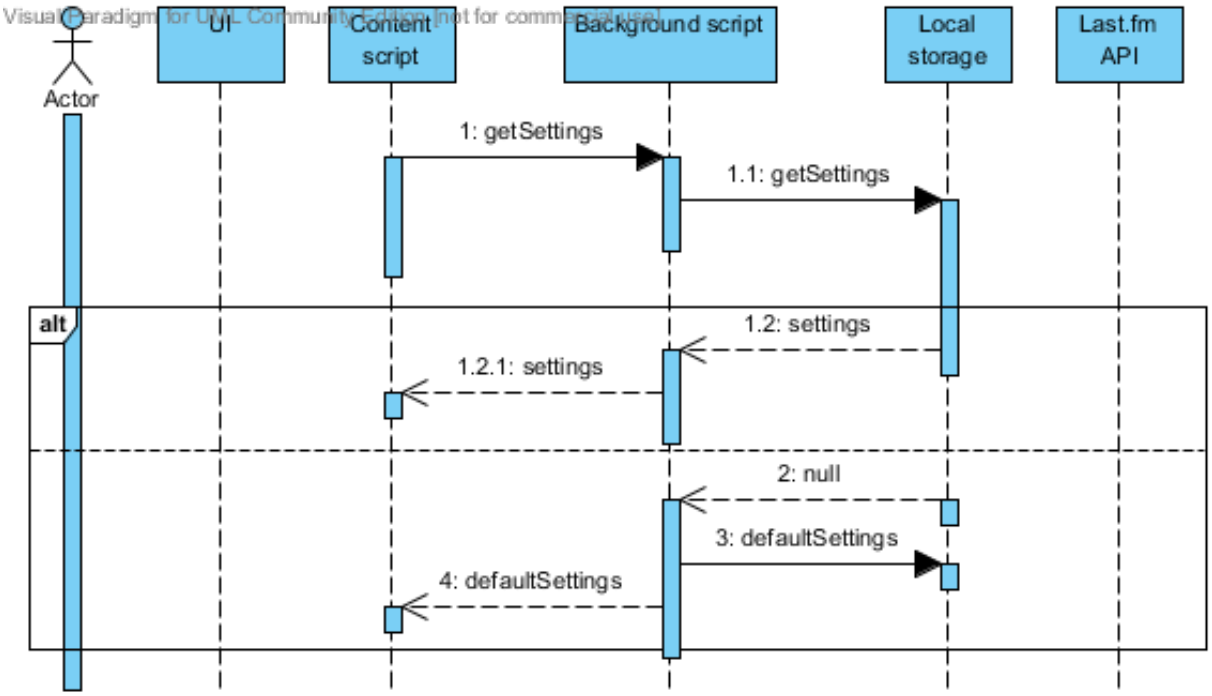


Figure 5.3: Sequence diagram: opening the Last.fm recommendations page part 2: retrieving stored settings.

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

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

Algorithm 1: Loading the data for the visualization.

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

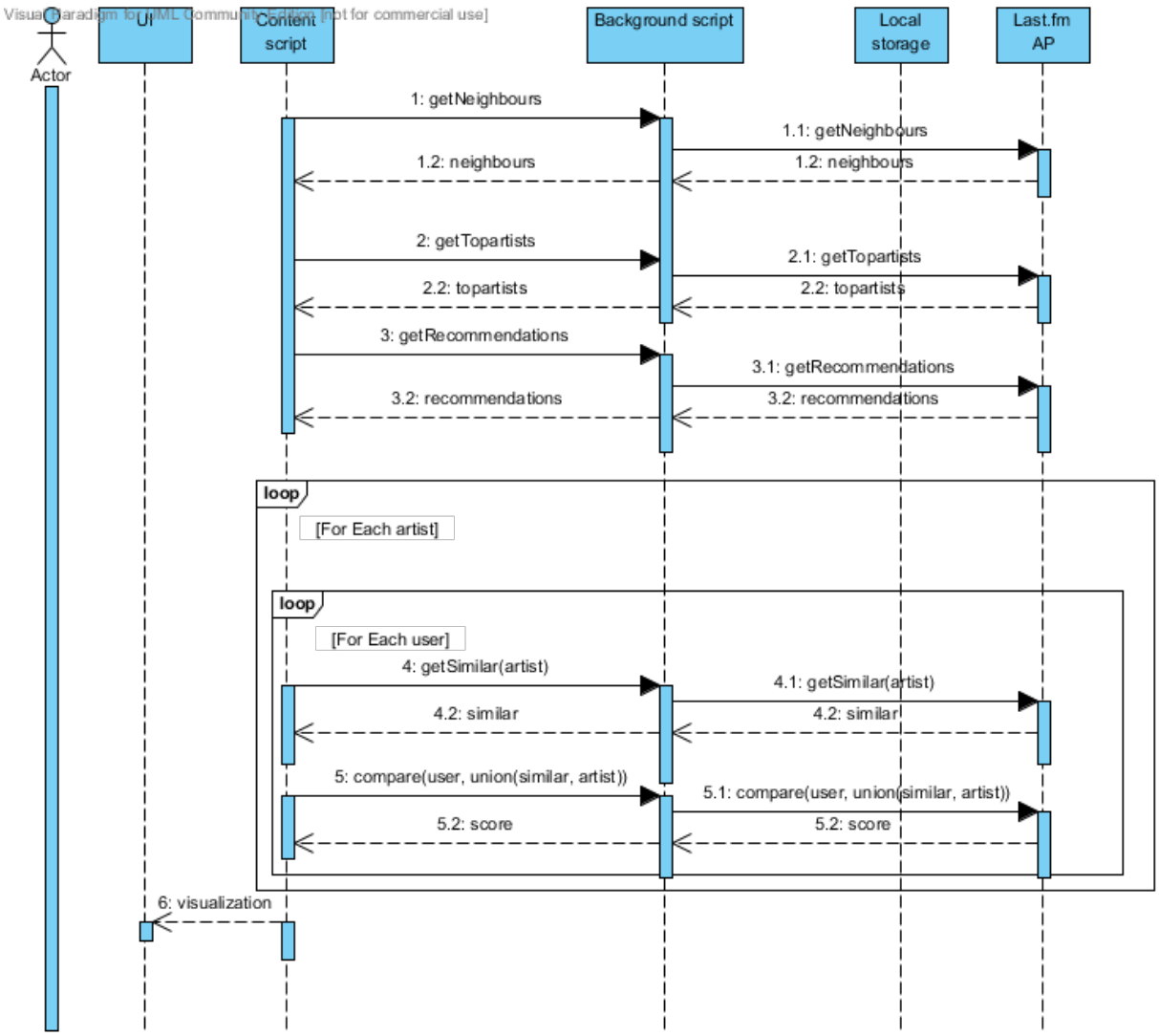


Figure 5.4: Sequence diagram: loading the visualization.

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

5.3 Implementation

5.3.1 Configuration file `manifest.JSON`

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

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

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

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

5.3.2 The visualization `infovis`

The visualization consists out of four main parts:

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

Data structure

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

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

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

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

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

Algorithm 2: Loading the data for the visualization.

The visualization script

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

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

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

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

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

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

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

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

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

Style sheet

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

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

Chapter 6

Conclusion and future work

Bibliography

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

[2] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, Dec. 2007.

[3] 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.

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

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

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

[7] 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].

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

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

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

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

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

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

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

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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].

[22] 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.

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

[24] C. Johnson, E. Hendriks, I. Berezhnoy, E. Brevdo, S. Hughes, I. Daubechies, J. Li, E. Postma, and J. Wang. Image processing for artist identification. *Signal Processing Magazine, IEEE*, 25(4):37–48, july 2008.

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

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

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

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

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

[30] 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.

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

[32] 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].

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

[34] 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].

[35] 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].

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

[37] 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.

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

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

[40] 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].

[41] 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].

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

[43] 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.

[44] 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.

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

[46] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 269–276, New York, NY, USA, 1993. ACM.

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

[48] 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.

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

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

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

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

[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] K. P. Tripathi. Article: A study of interactivity in human computer interaction. *International Journal of Computer Applications*, 16(6):1–3, February 2011. Published by Foundation of Computer Science.

[55] G. Tzanetakis and P. Cook. Marsyas: a framework for audio analysis. *Org. Sound*, 4(3):169–175, Dec. 1999.

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

[57] G. Tzanetakis, E. G., and P. Cook. Audio analysis using the discrete wavelet transform. *Proceedings of Conference in Music Theory Applications*, 2001.

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

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

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

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

[62] C. Ware and P. Mitchell. Visualizing graphs in three dimensions. *ACM Trans. Appl. Percept.*, 5(1):2:1–2:15, Jan. 2008.

[63] 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.

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

List of Figures

List of Tables

Index

Appendix A

SUS questionnaire questions

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

Appendix B

Task lists for the user tests

B.1 Paper prototype

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

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

B.2 Digital prototype

B.3 Working digital prototype

Appendix C

Quantified self

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