Recommender Systems in use

Postgraduate Student

Doctor of Philosophy
University of Edinburgh
2009

Abstract

My abstract.

Contents

Chapter 1

The Recommender Systems

1.1 Introduction

In our everyday life we are constantly exposed to recommendations in some form or the other. Examples of this are via televised commercials or signs in the supermarket. It is possible for the television and supermarket advertisers to make more personal recommendations by the use of demographics such as statistics of which age group that watches TV during certain hours and who buys certain items in which geographical areas. However, from the group of people that are exposed to these recommendations, it is possible that only a fraction will actually find the commercial relevant to their inherent needs or interests. What if there existed a way to target each individual user with personal recommendations? While we might have to wait a little with that in regard to the television and supermarket, it does already exist on the Internet and other places and are known as: Recommender Systems.
Where recommendations from the recommender systems described in this paper distinguish themselves from the above mentioned methods, is that the recommender systems are personalizing their recommendations for each of their individual users. This is done by acquiring information about each users preferences and interests, both given directly and indirectly by the user itself. Directly by the user typing in relevant information to the system and indirectly by observing the users behavior via the trail of technological footprints that is left behind when the user interact with a system. These footprints includes items bought, items viewed, what persons with similar interests have bought, what they search for, etc. This information is then used to compute the recommendations that can be presented to the user.

It is argued that we are moving away from an era of search based interaction and entering one of discoveryargued hvor? Find artikel der beskriver dette, en af de sidste vi har lst. The meaning behind this idea is that we are moving away from knowing what we want and hence search for, and instead are shown new products or sites that we did not knew existed, or that we even needed. It is likely that recommender systems will continue to evolve and keep increasing their efficiency, possibly until they can tell more about us than we on a conscious level can ourselves.

Research Question

In the world today, all users experience recommendations on a daily basis, as suggested above. Recommender systems are embedded into services like Netflix, Amazon, LinkedIn and many other places.
The purpose of this project is to get a deeper understanding of how recommender systems work and to suggest a recommender systems design for the IT-University, as well as explore the possibilities of designing with human values in mind.
The following questions will be discussed and possibly answered: How do recommender systems work? Do they all work the same way or are different types of recommender systems in play? What role does a philosophical perspective play in the making of a recommender system? However, the main purpose of this report are to answer the following:

What defines a good recommender system and how can a recommender system for courses at ITU be made?

We will try to answer this by surveying relevant litterature for recommender systems and explore the field of philosophy. This will be done by following the structure presented below.

Overview

This section are meant to provide the reader with a overview of the structure of the report.
The report is overall split into four chapters, which also reflects how the work process have been throughout the project.
The first chapter is about the different types of recommender system and how they work. This part describes two types of recommender systems and ways to make hybrid versions between the two.
The second chapter represents the philosophical part of the project. This section describes a framework, that we have tried to use in the design process for the recommender system for ITU.
The third chapter reflects our conception of the recommender system for courses at ITU. This part of the report have been produced by using the knowledge obtained from chapter 1 and 2.
Chapter four concludes the project and report and does therefore contain a discussion and conclusion of the proposed system and the project in general.

1.2 Collaborative Recommendations

The main idea behind this approach for recommendations, is to base the recommendations on users or items that are similar.
This method of recommending is one of the most widely spread, as it is used both at Amazon, Netflix and similar high-value companies (Linden et al., 2003).
In this section, the different implementation techniques of CF recommender systems will be described and a list of advantages and disadvantages will be given towards the end. We will distinguish between what is known as user-based and item-based recommendations (Jannach et al., 2014).
For both types of recommendations the basic problem can be formulated like this: For any given user and non-rated item, try to estimate the rating the user will give for this item.
The most common approach is therefore to define a user-item matrix as seen in the table below.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
UserA	3	1	2	3	3
UserB	4	3	4	3	5
UserC	3	3	1	5	4
UserD	1	5	5	2	1

Table 1.1: User-item matrix R

1.2.1 User-user-based recommendations

The approach described here are recommending on the basis of what is known as peer-users (also called neighbors), which means users that have similar preferences as the user the system is trying to recommend items to. The idea is: For item n, which *Alice* have not yet rated, find peer-users that have rated item n and based on this, compute the rating for item n for *Alice*. This means that the job of the RS is to: 1) find peer-users with similar taste to *Alice* and 2) take the rating for item n from the peer-users and based on this, predict the rating for item n for *Alice*.
To illustrate this, we return to the user-item matrix R in 1.1. Here *Alice* have not rated item 5 and the task of the recommender system is therefore to predict the rating for item 5, based on the ratings for this item from peer-users. Most commonly the peer-users are found by using the Pearson correlation coefficient(Jannach et al., 2014), which calculates the similarity between users.

Pearson correlation

Before describing the mathematical formulations, we first need to establish the basic terms required for the calculations. As described in section 1.2, a matrix of items and users are used. The users will in the following be denoted as $U = u_1, \ldots, u_n$, the items will be denoted as $P = p_1, \ldots, p_n$ and R for the $n \times m$ matrix for ratings r_i, r_j for $i \in 1 \ldots n, j \in 1 \ldots m$. The possible ratings a user can give is 1-5, where 5 being the the most-liked rating option. The similarity between two users a and b, given the matrix R is defined as follows:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r_a})(r_{b,p} - \bar{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r_b})^2}}$$

Doing this for each of the users and *Alice* will produce the similarity between *Alice* and each of the other users. The RS will then take the users with the highest similarity and proceed. In this example the most similar users are *userB* and *userC*. It should be noted that the formula takes into account that users tends to interpret a rating scale differently. Meaning that some users tend to give a lot of high ratings, whereas other never rates anything with a 5(Jannach et al., 2014, p. 15).

Now that we have found two users (B and C) that are similar to $Alice$, we are able to proceed and predict the rating for $Item5$, which $Alice$ have not yet rated. According to Jannach et al. (2014, p. 16) one possible way of making the prediction is based on this formula, where $\bar{r_a}$ is the average of $Alice's$ ratings, :

$$pred(a,p) = \bar{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r_b})}{\sum_{b \in N} sim(a,b)}$$

which produces the following:

$$pred(a,p) = 4 + \frac{0,85 * (3 - 2,4) + 0,70 * (5 - 3,8)}{0,85 + 0,70} = 4,87$$

Based on these predictions, we are now able to fill out the user-item matrix for $Alice$ and make a list of top recommendations. The list of recommendations is a product of both computed ratings, using the above method, and actual ratings from $Alice$.

The above presentation of a recommender systems is of course a very small example and in real world applications, the case of recommending is not as straight-forward as the method described here. The example just holds 5 users and 5 items, whereas real-world applications often contains tens of thousands or even millions of both items and users, as in the case with the Netflix Prize, which contained 480,000 subscribers and 18,000 movie titles (Bennet and Lanning, 2007). As we will later present, the problem of data sparsity with rating of items, is also a very present and problematic issue for recommendation systems.
This concludes the description of the user-user based recommendation technique. The second type of collaborative recommendations technique will be described next.

1.2.2 Item-based recommendations

Another technique of collaborative recommendations, called item-based, are in certain areas very similar to the user-based technique described above. The main idea of this approach is to calculate the similarity between items, instead of users, based on ratings for the items. If we want to compute the rating for $Item5$ and we return to the matrix in 1.1, we can see that all the ratings given for $Item5$, is close to all the ratings given for both $Item1$ and $Item4$. Because of this similarity, we can conclude the following: Where $Alice$ gave a rating of 5 to $Item1$ and gave a rating of 4 to $Item4$, an item-based recommender system will compute the average of these similar items ratings and give $Item5$ a rating between 4 and 5.

Cosine similarity measure

In the previous sections we looked at how $Item5$ had similar ratings with $Item1$ and $Item4$. Because of a small dataset, it was possible to observe this similarity without any kind of calculations. It is rarely the case that this is possible in real-world applications and one of the methods for calculating this similarity is to use the cosine similarity measure.
By making a vector for each of the items, based on their ratings, this method uses the angle between two vectors for finding their similarity. The formula for calculating this angle is found below:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

The cosine similarity values goes from -1, trough 0, up to 1, where 1 indicates a strong similarity between the two compared items, just like the Pearson method, as indicated on the picture below:

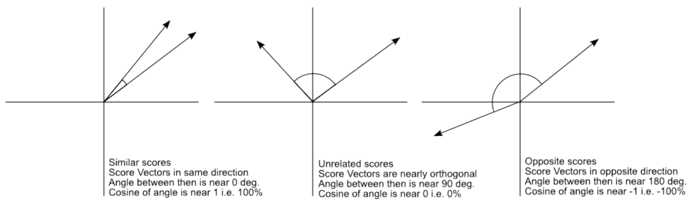
Similar scores
Score Vectors in same direction
Angle between then is near 0 deg.
Cosine of angle is near 1 i.e. 100%
Unrelated scores
Score Vectors are nearly orthogonal
Angle between then is near 90 deg.
Cosine of angle is near 0 i.e. 0%
Opposite scores
Score Vectors in opposite direction
Angle between then is near 180 deg.
Cosine of angle is near -1 i.e. -100%

Figure 1.1: source: http://pyevolve.sourceforge.net/wordpress/?p=2497

Differently from the Pearson method, this formula does not take into account that users use a rating scale differently. To take this into account, the *adjusted* cosine measure could be used:

$$sim(a,b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r_u}) * (r_{u,b} - \bar{r_u})}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r_u})^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r_u})^2}}$$

After the similarity between the items have been calculated, the recommender system can predict a rating for $Alice's\ Item5$, based on the ratings for the items similar to $Item5$. This prediction is done, using the formula below:

$$pred(u,p) = \frac{\sum_{i \in ratedItems(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItems(a)} sim(i,p)}$$

According to Linden et al. (2003) this is the specific method implemented in the web shop of Amazon.com. The reason for choosing this, over the user-user based approach, is for reasons of scalability. The user-user algorithm does not scale well, when the user and item space becomes large.

1.2.3 Advantages and disadvantages

Collaborative recommender systems is one of the most researched techniques of the different types of recommendation systems and because of this, many different types of collaborative recommendations exists (Jannach et al., 2014). For all these different types, it is possible to outline some common advantages and disadvantages, which will be done in this section.

One of the big advantages, compared to Content-based recommendations (which will be explained in section 1.3), are the fact that collaborative filtering works independently of the items it recommends. The only thing needed to produce recommendations are ratings, even if the items are dissimilar to those seen in the past (Adomavicius and Tuzhilin, 2005, p. 18). This does, however, also mean that the system are very depended upon ratings, which makes the system very vulnerable, if no ratings are given.
Another advantage is the possibility of adding new items. Compared to Content-based, new items can be added to the inventory without having to analyze the content of the item. But even though the process of adding new items are an easy and quick one, in order for the items to be recommended, it still relies on ratings.
Like with the Content-based recommender method, the collaborative method do also have some disadvantages described here as: new user problem and data sparsity.

New user problem, in common named cold start problem, are related to new users entering the system. This could be the case, if a recommender system were to recommend a movie to a user, which the system does not have any previous history of and the user haven't rated any movies. This is a general problem for both the Collaborative and Content based approach.

Data sparsity is a problem describing the lack of ratings within the system. Many recommendation systems suffer from this, where users tend not to rate items, and very often the available ratings are very small compared to the total number of ratings (Adomavicius and Tuzhilin, 2005, p. 19).

1.3 Content-based recommender systems

The overall reason to implement a Content-based recommender system is basically the same as for other recommender systems; to deliver a list of recommendations that statistically will be seen as valuable to the user. Where it differentiates itself from others is in the way these recommendations are generated.

A content-based recommender system will try to recommend items to the user similar to items that this user has interacted with in the past. In other words, the way the content-based recommender system approaches the recommendation process, is by analyzing previous items preferred by the user. From this information the user's interests are extracted and these will then be compared with items contained in the system.
The system will then return items that are seen as relevant for the user, based on his interests.

1.3.1 The Components

In this section we will describe the different components which represents the content-based recommender system. In the previous section we explained that it produces the recommendations in relation to the user's preferences, then by comparing these preferences with the items contained in the system, and finally choosing and presenting the recommended items to the user.
There are three main steps used in this recommendation process, which are described as follows; content analysis, profile learning, and filtering. Figure 1.2 illustrates the process and can be used as a reference to better comprehend some of the steps which are further explained below.

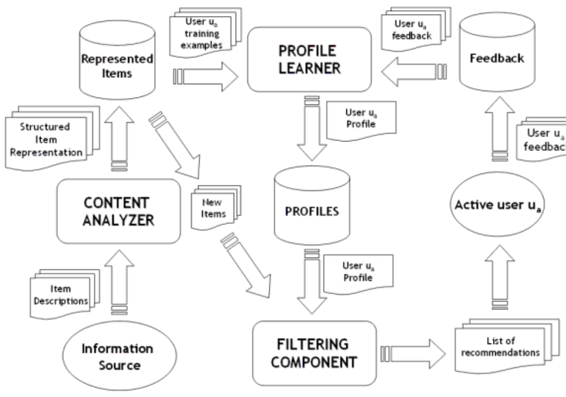

Figure 1.2: source: http://pyevolve.sourceforge.net/wordpress/?p=2497

- **Content analyzer:** The main responsibility of the content analyzer is to represent data (items, web pages, articles, documents, etc.) in such a way that it can be used by the next processing steps. This is done by shifting the content representation from its original form(usually text) to a form, represented as vectors, which is usable by the filtering compontent.
- **Profile learner:** This module collects and utilizes the relevant information that has been analyzed and altered into a usable format by the content analyzer. In this case, relevant information is data which represents the users preferences. The profile learner aspires to put this information in order and then use this ordered data to create the user profiles.
- **Filtering component:** This component receives items from the content analyzer and user profiles from the profile learner, both represented as vectors. The filtering component then compares the vector representing the user, with the vector representing the items, to compute and produce a list of recommendations.

These three components work together to recommend items to the user via a series of steps, as illustrated on figure 1.2.

The first step of the process occurs when the content analyzer receives a number of item descriptions from an information source.These descriptions are analyzed to extract specific keywords from the unstructured text, to end up with a structured representation of the items. In parallel to the proces of the content analyzer, the profile learner starts building the user profiles. These profiles can be computed purely based on information from the information source, but can also come from system profiles that users have created. These system profiles can contain user feedback, which can be either implicit or explicit. Explicit feedback refers to when the user actively makes a decision, such as giving a rating for a specific item. Implicit feedback refers to when the user does not make any active involvement, and is captured by monitoring and analyzing the user's activities and behavior.
The last step of the process in then for the filtering component to compare the user profiles from the profile learner with the items from the content analyzer. The filtering component produces a list of similarities between items and a specific user and prioritizes this list with the most similar items being in top.

In the next section we will discuss how the similarity of items in relation to user profiles are computed.

1.3.2 Vector Space Model

The way most content-based recommender systems defines the relevance of an item is by the Vector Space Model (VSM), which is a representation of text documents in the form of vectors. Each of the documents is represented by a vector in a n-dimensional space where n represents the number of keywords from that particular document that matches the overall list of keywords.
When using VSM to represent the weight of keywords in documents, there are two things that are important: 1) Giving a weight to the keyword, indicating the importance of the keyword in relation to the document, where the keyword is found in and 2) measure the importance of the document in relation to all the other documents contained in the system. A computational scheme which can be used for this is TF-IDF.

TF-IDF stands for Term Frequency / Inverse Document Frequency and is used to generate a weight for a specific keyword t_k in a specific document d_j, in relation to all documents. TF-IDF is divided into two parts.
The first part is Term Frequency which counts the number of occurrences of a specific keyword t_k in the document d_j, and gives this keyword a weight of importance in relation to the keyword in the document with the maximum number of occurrences. The second part is Inverse Document Frequency, which analyzes how many documents contains this keyword. The formulas for these calculations will be explained further below in this section.
If a keyword happens to be noted often in several documents, this specific keyword will not be given too much weight, because the system sees it as common. On the other hand, if a keyword occurs frequently in a few documents, but infrequently in all other documents in the system, this keyword will achieve a higher weight.

The document vector is generated by checking the keywords contained within the document. These keywords are given a weight in relation to their significance. The document vectors consists of the values of each keyword.

The user vector is generated by checking the keywords for the documents that the user has previously interacted with (bought, viewed, tagged, etc.).
The vectors of the document profile and the user profile are compared by the system by calculating the angle between the vectors. The smaller the angle, the more aligned the document is with the user interests. In many recommender systems, instead of the angle, the cosine to the angle is used, which gives a value from 0 to 1, 1 being the perfect match.

To understand the concept of the vector weighting we imagine that we have a content space, with as many dimensions as there a keywords in the system. Each of the documents and user profiles in the system are represented by a vector in this content space.

There are several ways where a vector can be computed. Here we will outline the three most common ways. 1) A vector can represent a fact (true/false) and can be computed simply by using a boolean value. An example of this can be whether a specific keyword is used in a specific document. A boolean would be used here because it is not possible to give this vector a further notion of intensity.
2) Another way to compute a vector is by counting the number of keyword occurrences in a document, which can help give a bigger notion of intensity for the vector.
3) Use the TF-IDF which implements both of the above mentioned methods, because it gives us a notion of intensity in a document along with a notion of distinctiveness for the individual vector. This approach is described below.

TF calculates a value for a specific keyword in relation to how many times it occur in the specific document d_j and how many times the most frequently used keyword k occur.

$$\text{TF}(t_k, d_j) = \frac{f_{k,j}}{max_z f_{z,j}}$$

The TF-IDF approach adds an extra layer to the TF, by further calculating the keyword occurrences in relation to all documents in the system.

$$\text{TF-IDF}(t_k, d_j) = \text{TF}(t_k, d_j) * log \frac{N}{n_k}$$

Cosine normalization is used to scale all the weights in the calculation to give them a value between 0 and 1.

$$w_{k,j} = \frac{\text{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} \text{TF-IDF}(t_s, d_j)^2}}$$

The cosine similarity is used to indicate the similarity between two vectors, which then can be used to predict whether a user will be interested in a particular item or not. The result is a value between 0 and 1, where 1 indicates a big similarity between the items.

$$sim(d_i, d_j) = \frac{\sum_k w_{ki} * w_{kj}^2}{\sqrt{\sum_k w_{ki}^2} * \sqrt{\sum_k w_{kj}^2}}$$

The methods described in this section are essential for a content-based recommender system because they are the foundation on which the recommendations are made.

1.3.3 Advantages and disadvantages

When looking at recommender systems from a content-based systems point of view there are several advantages over a recommender system in a collaborative-based form.

The three main areas of advantage are: User independence, transparency, and new items.
The **user independence** shows itself in that the content-based recommendations are solely based on the active user's own preferences and does not rely on the interaction with others.
The content-based recommendations are **transparent** in the way that all its recommendations are clearly based on the users list of preferences.
The advantage of a content-based recommender system in regards to **new items** that enters the system, is that these items will not need to be rated prior to being recommended to the users. Here, the only dependency is whether the item is similar enough to the individual users preferences.

As well as the above mentioned advantages, a content-based recommender system also has some disadvantages. The three main areas here are: Limited content analysis, over-specialization, and new users.

The **limited content** analysis revolves around the fact that content-based recommender systems most often are dependent on domain knowledge. An example of this could be for a movie recommendation. Here, the recommender system would need to know the actors, director, genre, and alike. If the content analyzer does not find this information within the provided data, then it will be virtually impossible for the recommender system to distinguish between items correlating with the user preferences and items that are of no interest at all.

The **over-specialization** is that the user will never be recommended any items in a positive unexpected manner. What is meant by this is that the user's personal area of preferences might be somewhat wider than the area of the current information held in the User Profile, but because the recommended items for the individual user is based on a similarity with items previously interacted with by the user, there will never be a recommendation deviating from this.
The **new user disadvantage** can be sort of a predicament for the recommender system. The issue here is that to give the new user a fair recommendation based on user interests and preferences, the user will need to have interacted with a certain amount of items for the filtering component to compare with. When the new user first create his or her account, the recommender system is somewhat forced to either recommend a set of random items or recommend what the average user would be recommended. In most recommender systems the latter is the solution of choice.

1.4 Hybrid approaches

In sections 1.2 and 1.3 we described collaborative and content-based recommender systems. These two have, of course, both advantages and disadvantages as also have described earlier. To overcome some of these disadvantages, one could combine the two approaches into a 'hybrid' recommender system.

In the following a list of applied hybridized approaches will be presented, to outline some of the different methods for doing this. According to Adomavicius and Tuzhilin (2005, p. 20-22) the methods can be classified as the following four: *"(1) implementing collaborative and content-based methods separately and combining their predictions, (2) incorporating some content-based characteristics into a collaborative approach, (3) incorporating some collaborative characteristics into a content-based approach, and (4) constructing a general unifying model that incorporates both content-based and collaborative characteristics"*. It should be noted here that this list is not comprehensive and many other hybridized approaches exists.
The first method is to implement the collaborative and content-based independently of each other and then obtain a recommendation for the same item from both, and then either combine the rating from each system into a single rating or select the 'best' rating relative to the recommended item based on a predetermined quality setting.
The second method is to add some content-based characteristics to a collaborative recommender system. Here the system uses the profile learner to compare the similarity between users and thereby partly overcome some of the sparsity problems, because user-pairs most often does not have many ratings in common. Another feature of this approach is that users can be recommended an item based on, not only if it is highly rated by peer-users, but also if this item matches the users profile.
The third method used some kind of dimensionality reduction technique on the profiles given from the 'profile learner' in the content-based recommender approach. In simple terms, this mean reducing the number of dimensions found in the vector space, in order to improve performance compared to the pure content-based approach.
The fourth hybrid approach is the combination of the collaborative and content-based into one single recommendation model. The approach have been adopted by many researchers in recent years and because of this, many different methods of this hybridizing approach have been proposed (Adomavicius and Tuzhilin, 2005, p. 22). These different methods somewhat lies outside of the scope of this report and will therefore not be further described.

1.5 Summary

This conclude our chapter on technical descriptions of recommender systems. In this chapter we have described recommender systems from a collaborative-based and content-based point of view, along with a brief description of a variety of ways to combine the two. There are many different shapes and sizes of recommender systems, and due to the time constraints on this report, it is very likely that we have only just scratched the surface of the world of recommender systems. It is very possible that there are numerous features that we have not been able to take into consideration. The next chapter will present the philosophical theory needed to design a recommender system.

Chapter 2

The Value Perspective

In recent years their have been a shift in focus within the design of technology. Designers and developers of information systems have gone from first realizing values in their systems when they reach the end-users, to now trying to foresee how systems can embed desired values during the design process.
An example of this, is the story of usability. Designers will recall that this value, in the past, was very much a desired featured, but was hardly given a focus in the actual design process. This lack of focus, often lead to information systems, which was rejected by end-users, even though the system fulfilled the specified requirements from stakeholders.
The observation here is that it is easy and straightforward to subscribe to the ideal of values (exemplified by usability) in information systems, yet a completely different story to actual use the values actively in the design process.

2.1 Value Sensitive Design

For reasons mentioned above, a greater focus upon supporting human values have to come to light and have emerged within four areas of design processes: Computer Ethics, Social Informatics, Computer Supported Cooperative Work and finally, Participatory Design(Friedman et al., 2006, p. 3). However, due to certain limitations within these approaches, another fifth approach have emerged, which:

> *"seeks to design technology that accounts for human values in a principled and comprehensive manner throughout the design process"* (Friedmann and Kahn, 2002, p. 1186)

This methodology is called Value-Sensitive Design and was developed by Batya Friedmann and Peter Kahn and the key components of this approach will be outline in section 2.1.
This design approach is not a big change in regard to traditionally engineering methods, like the Waterfall method or other developing methods, though there are some similarities between VSD and traditional methods. Many methods begin with some form of design phase, where key concepts are conceptualized and from here moves on to a analysis and development phase, which are somewhat similar to VSD's technical phase. The project typically concludes with some tests and an evaluation phase, similar to the empirical phase of VSD(Cummings, 2006, p. 704).
The VSD approach is meant as a tool and supplement for aiding system designers, toward a more structured way of incorporating ethical concerns and values into the design process.

The Value-Sensitive design framework has identified twelve specific values which is of great importance in the design of technology: Human welfare, ownership and property, privacy, freedom from bias, universal usability, trust, autonomy, informed consent, accountability, calmness, identity, and environmental sustainability. These values have been identified to have ethical importance and they should *"have a distinctive claim on resources in the design process"*(Friedmann and Kahn, 2002, p. 1187). Two points should be made about this list of

values. Firstly, all the values are not distinctive and could therefore overlap in certain aspects. Secondly, the list is not comprehensive, in the sense that many other values could possibly be identified to have importance in certain design projects.
In order to actively design with these values in mind, the framework are split into three parts, which enforces the value in different ways. The three parts are: Conceptual, Empirical and Technical and the following section describes these parts.

2.1.1 Conceptual

The first phase of the methodology is a philosophical investigation of the central parts of the system in question. Questions here include: "Who is affected by the system" and "How are certain values supported or dismissed by technological design".
The goal of this phase is *"... to identify one or two values of central interest that could be viewed as a common thread throughout the project"* (Cummings, 2006, p. 703). Even though it is possible that the identified values could be replaced by others during the project period, it is still important to realize them at an early stage, so the awareness of values consists for the entire project. Already in this phase should designers consider different things related to the proposed values. One question to ask, is whether or not a specific value *"... are universal to all humans or pertinent to local groupings only ..."* (Flanagan et al., 2008, p. 326). Designers may quickly agree upon a certain value being universal in their part of the world, say western society, but they still need to question this and maybe even conduct a empirical analysis to justify their initial thoughts.
Another important aspect of this phase is to identify stakeholders, which can be both direct and indirect. The distinction between the two, are how they interact with the system and how they are affected by the systems output. Along with identifying the stakeholders, a next step is to find benefits and harms for these and link them to values.

2.1.2 Empirical

The phase that follows is often the empirical investigation. This phase is a social investigation of the context of use and the systems impact on the involved stakeholders. Meaning that the focus here is on human interaction. A key focus here is the trade-offs between technical issues and competing values. Here the example of usability can be considered. If one imagine an Internet browser, which gives the user easy access to private browsing history, this gives third party software the same level of easy access, thus potentially compromising the value of privacy.
The phase is described to be used in two ways. The first is in relation to the design making process found in the conceptual phase. Here purely philosophical reasoning could be found insufficient, and one could therefore use a empirical investigating to establish which values would be most relevant in the particular setting. This would require designers to explore how the domain area works and how the identified values found in the conceptual phase, would impact stakeholders.
On the other hand, this phase could also be used to establish whether or not a certain value have been embodied successfully and if the system is used as intended. To examine the impact a certain value has in the use of a system, a empirical investigation can be conducted after the system has been deployed, by using traditional ethnographic methods of interviewing and observations.

2.1.3 Technical

The last part of the framework is a technical investigation. The focus is to explore how certain design decisions either support or dismiss the values identified in the conceptual phase. One question designers can ask is: how does this design choice affect the values identified in the conceptual phase?. The purpose of this phase is therefore to resolve what kind of values the technology itself is about to embody. Even though the technical and empirical may seem similar it is important to realize that they are not, like described by Albrectslund (2006, p. 67):
"It should be noticed that technical analysis differ from empirical analysis in that the former focus on technology itself, whereas the latter focus on people and social systems influenced by

technology"
This means that the two phases have entirely different points of interest. The empirical phase looks at the social implications by a technology, whereas the technical phase focuses on the technological mechanisms that can support the chosen values.

Chapter 3

The ITU Recommender System

So far the report have mainly been focused upon the different types and methodology of recommender systems along with the theory for values in design. At this point, the reader should have been provided with an overview of the two topics. In this third chapter the objective is to use and apply the obtained knowledge from chapter 1 and 2 to design a recommender system, with some desired values in mind.
A recommender system can be seen in many forms and be found many places, as described in chapter 1. For this report, a recommender system for courses at the IT University, will be conceived. The reason for this, is that all full-time students are faced with one or more options for non-mandatory courses throughout their education. This can, for some students, be hard to do, because of the many courses available as well as the lack of transparency of the content for each of these courses.
It is important to note that the proposed system will not be build, but only be designed at a fitting layer of abstraction. The most natural step would of course be to create the design along side the development of the system. There are two reasons for not doing this. The first is a pure time related issue. This project was done over a period of approximately 3 months, where the primary focus has been to read the needed literature to present a fitting perspective upon the topics of recommender system techniques and values in technology. The second reason for not including an implementation stage are grounded in the process. A popular thought in these times are the one about agile and iterative development methods. Here developers wants to move away from the more traditionally waterfall method and into a more agile method, which in some cases means coding while designing. This is a natural process, which converts the design decisions into code almost at the same time. This way there is a short way to prototypes. To do this, the design process has to reflect the levels of details needed to implement the decisions. The code will then end up reflecting the designed intentions, but at some point it is very likely that the code will begin to impact the design process, which is exactly why this methodology is so strong. Design and code is correlated and influences each other. Because of the time restraint, we believed the most appropriate approach, would be to focus only on the design process and exclude the implementation This way we would be able to work in a linear process and focusing on the best way to embed the desired values, without having to worry about the details of the implementation.
This means that the following section will go as much into detail as possible for the design process, without worrying about the low level details of implementation and production. With this in place, we will continue with the design of the system.

3.0.4 ITU Domain

To design the system it is needed to explore the domain in which the system will be designed. To get an overview of this, this section will explain the different terms which are used and an explanation of what kind of educations the IT University offers their students.
The educations is divided into 3 sections: Bachelor, Master and Diploma. Both the Bachelor and Master are full time studies, whereas the Diploma is targeted for people who are already working and wish to complement their education. There are three types of bachelors tracks: Global Business Informatics (BGBI), Software Development (BSWU) and Digital Media and Design (BDMD). The Masters are divided into: Digital Innovation & Management, Digital Design & Communication, Software Development and Technology and lastly Games. Under the Master programs the students can choose different tracks (also called specializations), but these will not be taken into consideration in the system, and are therefore not explained here. The Diploma is designed to be a part time education, with single elective courses. Here it is also possible to choose tracks, but like for the Masters, these will not be further explained. A bachelor education consists of 180 ECTS points, a Master of 120 ECTS points and a Diploma of 60 ECTS points. Since the Bachelors consists of many mandatory courses, not a lot of ECTS points are available for elective courses. The direct opposite is the case for the Masters. Here most of the 120 ECTS points are elective, which is the same for the Diploma.
The different people associated with ITU can be divided into three groups: Students, Researchers and Staff. The most interesting group is of course the students, because this is the group for which the recommendations will be targeted. This group can be further divided into full- and part-time students. Most people on the Bachelors and Masters programs are full-time students, whereas a small part of Masters as well as all people on Diploma are part-time for various reasons.

3.1 Conceptual

In accordance with the VSD-framework, this section describes how we practically identified the stakeholders for the system along with the benefits and harms the system can cause for these stakeholders. We will continue to define these benefits and harms and how they relate to the values.

Stakeholders consists of individuals or groups that are directly or indirectly interacting with or are affected by the system. Direct stakeholders are those that are interacting with the system in a direct manner or are subsequently affected by or interacting with the output of the system. Indirect stakeholders are people who are affected by the system though they never directly interact with it. It is possible for an individual to be both directly and indirectly affected by the system, and hence being both a direct and indirect stakeholder.
Benefits and harms defines the good and bad sides that the system have for the stakeholders of the system. According to Friedman et al. (2006, p.12) it is described that it might be possible in some way to make every human an indirect stakeholder of the system, but here stresses the importance of prioritizing the individuals or groups that are most affected on an indirect level.

3.1.1 Stakeholders

To find the stakeholders for the ITU recommender system, we sat down and tried to identify all the people that were interacting with or affected by the system in any way. Below are the results we came up with.

Direct stakeholders

- Students are interacting with the output of the system
- Teachers are writing and changing the course descriptions and will therefore have opportunity to alter the output of the system
- Persons that will be in charge of determining the keywords for the system.

Indirect stakeholders

- Teachers are indirectly impacted in regards to whether students choose to sign up for their course(s) or not, because the course can potentially be canceled if there are not a minimum number of sign-ups.
- Teachers assistants are impacted indirectly the same way as teachers.

The above show us that the teachers and the students that are teachers assistants are all direct and indirect stakeholders of the system . As it can also be seen, we did not find a great deal of different stakeholders, and we suspect that this could be because the amount of direct interaction with the system is very limited.

3.1.2 Benefits and harms

This section describes the benefits and harms for the system's stakeholders. We recognized the different benefits and harms through a discussion following the definition of our stakeholders.

Benefits for direct stakeholders

- Students benefit from being given suggestions on behalf of their previous selected elective courses, which can support them in finding their next elective course(s).
- Teachers benefit by being able to specify very precisely what the course is about and then getting students which prefer these preferences.

Benefits for indirect stakeholders

- Teachers benefit of students being recommended their courses, which will potentially help reach the minimum number of students of the course
- Teachers assistants benefit from the same as the teachers does, in that when the minimum student limit is reached, they have a job for the following semester.

Harms for direct stakeholders

- Students who do not have any previous elective courses, will give the system no information to base the recommendations on.
- Teachers They can potentially be harmed by the fact that they themselves have to update the course information. A poorly course description might lead to fewer recommendations of this course. They can also be harmed if the producers of the non-keyword list adds a non-keyword that is essential to one or many of the teachers courses, since this will result in the course in some cases will not be taken into account.
- Persons responsible for the keyword list, can accidentally add a wrong keyword to the list, which will change the outcome of the recommendations. This can give the person a bad reputation.

Harms for indirect stakeholders

- Teachers could experience students joining their course, based upon a low quality recommendation. This could make the student negatively influence the class sessions of the course.
- Teachers assistants could experience the same as the teachers.

As can be seen above, there are a variety of both benefits and harms that can affect the stakeholders of the system. In the next section we will have a look at our chosen values.

3.1.3 Values

This section describes the values that we have chosen as the primary values we will strive to embed into our ITU recommender system. The three primary values we have chosen to focus on, are freedom from bias, relevance, and diversity. These values have been chosen based on the benefits and harms described in the previous section. Beneath we have described what the meaning of each value is in relation to the ITU system.

Freedom from bias

The meaning of freedom from bias is for us simply that the system we are developing does not contain any biases. To assist us with that in the development process, we have used some of the research from insert reference on her framework about biases to get a better understanding of how to accomplish this. The reason we have chosen this value is that in our experience of developing software projects, the developers all have an initial idea of how a user will use and interact with the system. These initial ideas can be correct, but it is also likely that the developers are somewhat biased in their perception. Our thoughts are that if these assumptions are not tested in direct relation to the users, the system has a risk of being biased. Freedom from bias is the overall value we have focused on for our project, because we believe that it is an ideal value to strive for in any system regardless of context. We have therefore chosen to go a bit deeper into the aspects of this value than the others. In order for us to better understand how to prevent the implementation of biases into the system, we will describe the three different types to be aware of in relation to insert reference framework.

- **Pre-existing** bias describes a bias that in relation to the system exist by itself and in general before the system is created. A pre-existing bias can evolve from several different places, some being of societal character, subcultures within a society, or via private or public institutions and organizations. They can furthermore be a reflection of personal biases and opinions from people closely related to the design process of the system, here putting a high emphasis on the system designers. There are several areas from which a pre-existing bias can arise. An important thing to remember is that these can show themselves via a conscious effort from the people involved in the decision making process behind the system, but equally as well be integrated on an unconscious level, even when the system designers are trying their best to consciously avoid doing so.

- **Technical** bias describes a bias that is formed when system designers are trying to resolve technical constraints. These biases can occur in relation to both hardware and software. Hardware-wise the biases are getting fewer, one reason being responsive web pages that throws away the need for specific monitor sizes. Software-wise the biases can occur in a variety of ways. For example algorithms that does not uphold a level of fairness to all users. This can be an algorithm that is used in the wrong context, for example if the system is designed to perform a specific task, and then at some point is used to solve a different kind of task.

- **Emergent** bias can only arise in the context of use of a system. The emergent bias can show itself some time after a design has been made. One place that seems to be prone to emergent biases is in the systems where user interfaces play a vital role. User interfaces are most often designed for a specific set of users. New users might not be able to grasp the system functionality in the same way as the originally intended users, thus creating an emergent bias in the system.

The above points fairly describes the types of biases to be aware of when developing a system and the next describes the value of relevance.

Relevance

The meaning of relevance for us in relation to the ITU system, is that the students will be given recommendations that are somewhat relevant to them, both on a personal as well as professional level. The personal level here indicates that the recommendations will be aligned with the students personal preferences in regard to previous chosen courses. The professional level indicates that the recommendations will have relevance for the students line of study.

We have listed some benefits and harms of implementing this value into the ITU system.

Diversity

Diversity as a value in the ITU system means that even though we are trying to give as precise recommendations as possible in relation to the previous elective courses chosen by the student, we still want to bring an element of surprise into play, and possibly recommend a course that the student could be interested in, even though it lies outside the students previous preferences. The level of diversity we will embed will stay inside the frames of the students line of study, to still maintain a high level of relevance.

These above findings concludes our conceptual part of the VSD framework approach. The implementation of the values will be further elaborated in the technical part of this chapter.

3.2 Empirical

According to the VSD framework, a part of embedding values into a system is to investigate what values the system should pay attention to, examine how these values are treated by future users of the system, and to see if the intended values do in fact 'impact' the users and intended. This means that the phase could be used both before the technical phase, but also after to see how the intended values are 'received' by the users.
Due to the time constraint described in the introduction to this chapter, this phase have not been completed. Instead, for this project, the phase have been used to try to imagine what questions and areas that needed answer in order to design the system. This have lead us to two topics that needs investigation: 1) the ITU domain and 2) the intentions and thoughts behind the courses, educations and tracks. The first is a purely a questions about understanding the domain that the system needs to be produced in and for. Questions like "How many bachelors lines are there at ITU and what do each line hold?". The second is more related to the purpose of courses and different lines. A question here could be "Why does the bachelor lines have both elective and mandatory courses?"

This section could be about some data gathering from leading people from ITU, who has something to say about the student profile, when graduating from ITU. It could also be something about what leading people from ITU says in general about the students choice of optional courses and the purpose of having optional courses at ITU.

3.3 Technical

This section describes the actual domain where the system is to be designed and the technical choices taken for the systems. This will be done with the chosen values from the conceptual section in mind. The section are divided in to two parts. The first will be an analysis, whereas the second will focus upon the implementation and challenges for doing so.

The intuition for the system can be divided into three basic steps. The first is some sort of data extraction from the systems at ITU, that holds the data needed to make a recommendation. The second step is to prioritize this data and select only the relevant. The third step is to process the selected data and transform this into some sort of structure holding students and courses and thereby constructing the recommendations.

3.3.1 Design Choices

This section presents the design choices made throughout the design process. We will begin with describing the choice between different types of recommender systems.

Collaborative or Content

One of the first and most important choices to be made for this system, is what kind of recommendation type that should be used. In section 1 we outlined three different types of recommender systems: Collaborative-filtering (CF), Content-based (CT) and Hybrid approaches. As also explained in 1.4 it is most common to use a hybrid of the two. To keep the system as simple as possible for a start, the approach chosen is a content-based one. The reasons for this, will be outlined below.
The CF approach mostly builds upon ratings from users, which is used to either compare users with users or items with items, and then produce a list of recommendations. This makes the ratings a central and crucial part of the system. In order to implement this approach for a ITU recommender system, these ratings could come from the course evaluations, which basically asks the students to 'rate' and comment on both the course and teacher. This could definitely be a viable solution, except that it relies a great deal upon the data from the course evaluations. Unfortunately, the evaluation is only done by around 35-40% of all students, which means the system will be affected by the 'data sparsity' problem described in section 1.2.3. Even though many recommender systems suffer from this problems and are still used, the other approach have been chosen.
Another problem would be the reliability of the ratings. Even though these would come from the students, and thereby represent their opinion about the courses, these would properly be more of a reflection how popular a course is, more than the actual content of each course.
The primary reason for choosing the CT approach is the availability of keywords for each course. Each of the courses available for the students has an associated course description, which will be the primary source of content. In contrast to the CF approach, using the keywords from the course description instead of ratings, would represent what the teacher thinks and imagines about the course, which is a more authentic and representative source of information in our opinion.

Persistent or memory

As part of the design process, it is necessary to look at the usage of the system, in order to choose if data should be read into memory at run-time when needed or a more persistent choice of data storage should be used. The answer to a question like this is closely related to when and how often the system is running. If the recommendations are needed many times daily or weekly, the choice would be to have the system running on a server, with a database taking routine backups of things like the users profiles and recommendations. If the the system only would run a couple of times a month or year, one option would be to load all the raw data into the system for each run. Even if this process would be expensive to perform, it is easy to argument this being okay, since the data would only be used a few times and could be loaded in before actually needing it. Another option would be to load the raw data into the system on the first run, stripping it for unnecessary information and then storing this data in a database and then only update this database with new data, before each run. This latter option have been the chosen solution, since it enables the system to spend less time reprofiling all students for each run. It is only necessary to profile new students and courses and remove courses that are no longer available. This should in theory reduce the time needed for calculating the recommendations.

3.3.2 Architecture Analysis

In order to give the reader an overview of the system, this section presents the components involved in the system and outline the overall flow of the program. The diagram below displays a view of the system with its components:

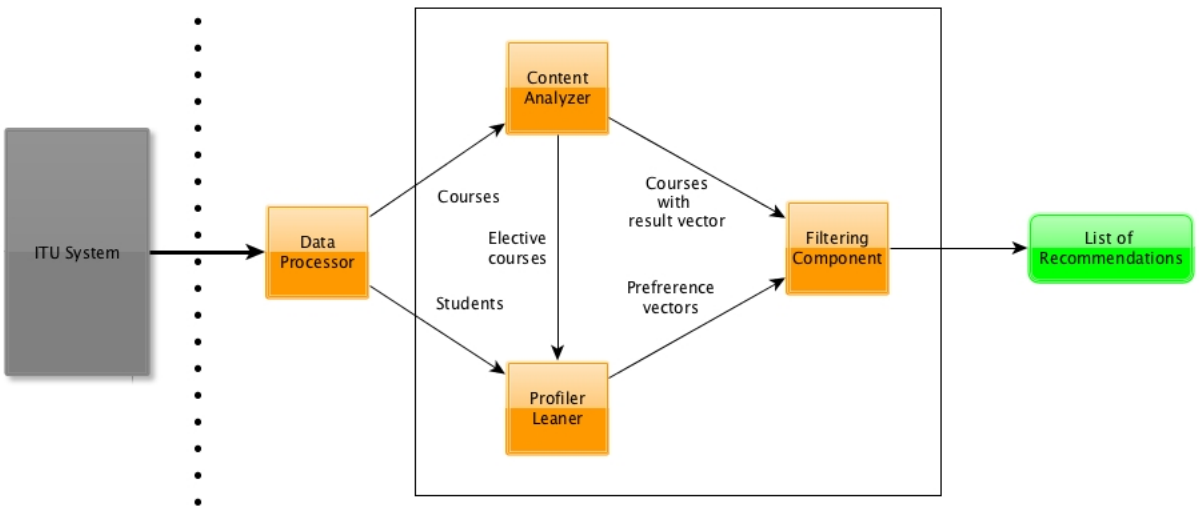

Figure 3.1: Architecture-Diagram

To the left the black box of ITU is displayed. This represents the data source, that will be used in order to produce the recommendations. The data will consists of all information about students and courses. The data processor component will receive all this data and sort it out, keeping only the relevant information. The data will then be split among the content analyzer and the profile learner, which will construct data structures of courses and students, as explained below. The filtering component will then receive these structures and output the lists of recommendations for each students.

3.3.3 ITU Recommender System

So far the previous sections have described the domain and the surrounding parts of the actual system. In this section, we will go in to depth with the actual recommender system and its individual components within the square in figure 3.3.2. The responsibility as well as the input and output will be described for each components. The order of this section follows the flow of the program and will therefore start with the content analyzer.

Content analyzer

This component are responsible for analyzing all courses available at ITU and construct a vector for each of these courses. The input of courses, will come from the data processing component and the output will be a in some sort of abstract data type, possibly like a map. The process of producing the vectors are expressed in the pseudo algorithm below:

- Make a list of non-keywords like "this", "and", "is" or "will".
- Construct a string parser which identifies all words in the current document and applies a hash-function on each of these words, except the ones from the non-keyword list.
- Count number of occurrences of each word found by the string parser.
- Repeat step 2 and 3 for all courses and construct a data structure of maps with maps. The first map has the course code as key and the value being a map containing each specific keyword as key and the number of occurrences as value.
- Use the TF-IDF formula for each course, which will produce a vector in a space, which contains n-keyword dimensions. The vectors represents the weight of all keywords for a course.
- Use the Cosine normalization formula for normalizing every keyword vector for all courses.

At the end of this algorithm, each course will have an associated vector. This vector represents the weight, in relation to keywords, for the course compared to all other courses. The resulting data structure, will then be used by the filtering component and the profile learner, which will be described below.

Profile Learner

This component is used to analyze the students preferences. To do this, some feedback or history from the students are needed. This feedback will come from the previous elective courses, that the student have been signed up for. The reason for using this information, is the fact that we believe that this is the most significant and reliable source of information. Like the content analyzer, the flow of this component are expressed in form of a pseudo algorithm outlined below:

- Retrieve all elective courses the student have been signed up for, from the content analyzer.
- Of these courses, compute the average vector (adding the vectors together and divide by number of vectors) by projecting each of the vectors on each other. This resulting vector represents the users preferences, in relation to the elective courses the student have signed up for.

If the student have had no prior elective courses, the profile learner will notify the filtering component, which will decide how to proceed.

Filtering Component

This component receive the results from the content analyzer and the profile learner, in order to produce the list of recommendations for each student. The input needed, are the courses with an associated keyword vector and a vector representing the preferences for each student, based on previous elective courses. For new students, this information will not exists. In this case the filtering component will produce a list of random courses, but only out of those courses, that are 'relevant for' (which is a tag placed on each course) the students line of study. When the student have had elective courses the component generates the list of recommendations like described below:

- In the list of courses taken from the content analyzer, it sorts out all the courses the student have already been signed up for.
- In the list of courses received from the content analyzer, sort out all courses that the student will have as mandatory courses in the future.
- Use the resulting vector produced in step 2, to compare every course from the content analyzer to produce the list of recommendations.

- Sort the list of recommendations, prioritized by courses with the most similarity (closest to 1) with the resulting vector

The output of this component will be a list of recommendations sorted after the courses which vector is the most similar to the vector representing each students preferences. This list will then be shortened down to an appropriate length with courses and sent out by email to every student.

3.4 Summary

This chapter describes a possible way of designing a recommender system for the courses at ITU and describes the outcome of the three phases of the VSD framework.
This section will reflect upon some value considerations and describe some possibilities for future improvements of the system.

3.4.1 Value Considerations

The Bias perspective

As mentioned in section 3.1.3, we believe that freedom from bias is a value that represents an ideal that one should strive to embed into any system, but also emphasize that it is hard to be completely 'bias free'. This section will outline an example of each of the three different types of biases and how we have tried to limit their implications in the ITU system.

The selected courses
In our suggested system we are ignoring the case in which students have been previously signed up for a course, but failed that course. The system will still see this course as attended by the student, and the course will therefore be sorted out, before the recommendations are produced. This represents a technical bias, since the system is not taking these courses into consideration, even though they are still possible to attend by the student. The reason for this decision is that we feel that a student who has attended a course earlier, is in a better position to decide whether or not this course will be a priority once more.

The list of keywords
The rationale behind the list of keywords, was initially based upon the idea of getting educational leaders found at ITU to produce a list of keywords, they found to be of importance in a system like this. However, we realized that this approach could possibly lead to a keyword base containing pre-existing biases. The reason for this is that the leaders, most likely unintentionally, all would believe that keywords related to their study were the most important.
This could be solved by allowing all keywords to be added to the list, though this would raise another possible predicament, since one line of study might wish to include several keywords while another only a few, which could tip the balance of the recommendations to one side.

In order to limit these issues with pre-existing biases in the keyword base, we decided that a person with no direct relation to the system should create a list of words that were not to be included in the search. By using this approach we ensures that all keywords are matched, except the ones found in the non-keyword list, thereby making the system less biased. This also makes the job of maintaining the system easier, because it only relies upon the non-keyword list and course descriptions.
However, if a teacher obtains some knowledge about this recommendation process, he or she is able to edit the course description in a way, that can alter the analysis of keywords, and thereby the outcome of recommendations.

The user interaction
In a typical recommender system, users a able to create profiles as well as provide explicit feedback for items. Users of the ITU system does not have these options, since no user interface are provided. They only receive an email with a list of recommendations. This have been chosen in order to limit the possibilities of a emergent bias, even though this type of bias can be hard to foresee in any type of system.

Diversity and Similarity

In the start of this project we imagined a recommender system based upon diversity, opposite to typical recommender systems, which are designed upon similarity.
We discovered that this approach is very much related to the context of use, and in the context of ITU it was not applicable with a recommender system built upon diversity. The reason is that similarity ensures a higher level of relevance in relation to the users preferences, which we

believe will be more beneficial for the students. However, we have chosen to design the system in a way where a bit of diversity have been implemented as well, in the form of a limited number of randomized recommendations. To still keep a level of relevance in these recommendations for the student, they are chosen from a list of courses, where the tag 'relevant for' equals the students line of study.

3.4.2 Future improvements

Feedback from users

We know that in order to make considerate elective course recommendations for the students, the system will need some information or feedback from the users. At the moment the only available feedback is in the form of previously selected courses. This creates a problem, when a user has not had any elective courses.
The way of improving this, could be by using the history of former students on the same line of study. In other words, it could be done by cross referencing line of study with chosen elective courses from former students, and hence give an indication of which recommendations might be relevant and useful for the student in question.
We acknowledge that this approach will make the system slightly biased in relation to new courses which have never been previously selected, and therefore will not be included in these recommendations.

Seat Reservation

As of now, the ITU system recommends courses that students might find interest in. A possible addition to this system could be in the form of seat reservations.
This would mean that instead of just recommending courses to students, the system would include the number of seats available in a certain course as well as reserving a seat in the course for which the student who were given a recommendation for. This would, in our opinion, increase the level of service provided by the system, since the list of recommendations would contain more relevant information in form of available seats and a reservation would be made for the student. This reservation should then last only a specified amount of time, to eventually free up the reserved seats.

Chapter 4

The Final Words

4.1 Discussion

The Process

The intention of this project was to investigate the areas of recommender systems while also obtaining knowledge within the area of ethics and philosophy.
The first part of the project period was used to examine the area of recommender systems. Here we quickly discovered that the area was more complex than we initially assumed. This meant that we had to limit the scope of the study of recommender systems in order to make room for the area of ethics and philosophy, which was explored during the second part of the project.
Where the area of recommender systems are within the field of mathematics, and thereby our line of study, ethics and philosophy are not. This meant that this area was a particular challenge, because we not only had to examine a great deal of literature, but also understand the language and terms used within this field.
From the beginning, we had an idea that both of these areas would be a challenge to cope with, but we were still surprised with the amount of effort needed in order to comprehend and use both areas in a sensible way in the project. This have resulted in a report which reflects the overall meaning of both areas, but does not present them with the depth that we had initially hoped for.
A better approach could have been to explore the possibilities of the VSD framework within a already known area of software engineering, instead of having to merge two areas, of which we had no prior knowledge, within a project of only 3 months.

The 'best' recommender system

As a starting point for this project, we based the research question upon what a good recommender system is. It is hard to describe what the 'best' recommender system looks like, because the design of a recommender system will always depend on the context of which the system are to be implemented. It is however possible to mention some things, that we believe will help in defining a good recommender system in general.
For one, a recommender system should not be imposed with any forms of biases. According to section 3.4.1, this is not always an easy task to accomplish.
Even though we had the value of freedom from bias in mind throughout the project, we still ended up with some forms of biases. Yet, is it still something that designers should strive to limit as much as possible.
Another thing is the level of relevance. This is very context specific, but a recommender system must always seek to heighten the level of relevance as much as possible. In the case of the ITU system, this have been done by only using an authentic source of feedback in form of history of selected courses.
We could have chosen to include feedback in form of student replies in the course evaluations and possibly giving them the option of rating the courses that they have been recommended,

in order to receive a new list of recommendations. These two options of feedback was not included in the system, because we felt that this feedback could be inconsistent, since not all users participate in the course evaluation.

The ITU recommender system

The second part of the research question found in section 1.1 asks how a recommender system for courses found at ITU can be made. We have proposed a design for a system in chapter 3, which reflects the use of the VSD framework together with the knowledge of recommender systems.
Due to reasons explained in section 4.1, the design of the ITU system are not as complete and comprehensive as we would have liked it to be.
On top of these, designing a system without producing the code in parallel with the design, makes it hard to realize how the design choices affects the actual implementation of the system. In our opinion, these two things have had a severe impact on the design of the ITU system.

Having said that, we do believe that we have reached our initial goal of making a draft for a design, though there is still room for improvement.

4.2 Conclusion

We conclude this project and report with this final section. We have described important aspects of what we believe a good recommender system at minimum should include and suggested a novel design for how a recommender system for course found at ITU could be made.

By the literature that we have surveyed it seems that an increasing focus upon Values in Design have begun to emerge in various fields of research. We recognize the great importance of designing with values in mind, which is what we have tried to do throughout this project. Along side this process, relevant literature of recommender systems have been used for the design of the ITU recommender system. While this have been a challenging task, we feel that we have obtained relevant knowledge about recommender systems in general and that we are able to take this knowledge with us into future projects related to Information Retrieval or similar areas.

Chapter 5

The Appendix

Bibliography

Adomavicius, G. and Tuzhilin, A. Towars the next generation of recommendender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions*, 7(6), June 2005.

Albrectslund, A. Ethics and technology design. *Ethics and Information Technology*, pages 65–72, 2006.

Bennet, J. and Lanning, S. The netflix prize, 2007. URL `http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf`. [Online; accessed 10-04-2014].

Cummings, M. L. Integrating ethics in design through the value-sensitive design approach. *Science and Engineering Ethics*, 12(4):701–715, 2006.

Flanagan, M., Howe, D. G., and Nissenbaum, H. Embodying values in technology: Theory and practice. *Information and Moral Philosophy*, pages 322–353, 2008.

Friedman, B., Kahn, P. H., and Borning, A. Value sensitive design and information system. *Human-Computer Interaction in Management Information Systems: Foundations*, pages 348–372, 2006.

Friedmann, B. and Kahn, P. H. Human values, ethics and design. 2002.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. *Recommendender Systems, An Introduction.* Cambridge Unversity Press, 2014.

Linden, G., Smith, B., and York, J. Amazon.com recommendations - item-to-item collaborative filtering @ONLINE, February 2003. URL `http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf`. [Online; accessed 02-04-2014].