# Twilm

**PROJECT**

---

# ONELINER ABOUT

---

*Authors*
Martin Christian Havig

*Supervisor:*
Some One

March 12, 2014

**Acknowledgments**

**Abstract**

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Contents

## 1.1 Purpose

## 1.2 Motivation

In today's day and age the increasing amount of data overwhelm our human processing capabilities in many information seeking tasks. To cope with this overload researchers have introduced recommender system to filter the ever increasing information and only present a small selection of items which reflects the users tastes, interests and priorities. Recommender systems are an active research field and has been successfully applied to many different systems ranging from e-commerce sites such as *Amazon*, movie and TV-series streaming services like *Netflix* and in different music applications such as Last.fm and iTunes.

Many/most/all of the largest commerce Web sites have been using recommender systems to help their customers find products to purchase for nearly two decades. Schafer et. al. [5] identified three ways, in which recommender systems increase E-commerce sales: (1) Browser into buyers: Recommender systems can help customers find products they wish to purchase, (2) Cross-sell: Recommender systems improve cross-sell by suggesting additional products for the customer to purchase and (3) Loyalty: In a world where the competitor only is one click away, gaining customer loyalty is an essential customer strategy. Recommender systems improve loyalty by creating a value added relationship between the site and the customer.

## 1.3 Context

| Chapter | Description |
|---|---|
| Chapter 1 | The Introduction chapter gives an overview of the project to the reader. It also outlines the purpose and motivation of the project. |
| Chapter 2 | The Preliminary Study chapter documents knowledge, research and technology that is relevant to the project, and how and why some of them were prioritized over others when it comes to how they are used in the project. |
| Chapter 3 | The Requirements chapter describes the requirements of the project. It also describes how and why they were created. |
| Chapter 4 | The Design chapter describes the design of the system and how it was made. |
| Chapter 5 | The Implementation chapter describes the implementation of the system. |
| Chapter 6 | Evaluation chapter discussed the development process, testing of results and major issues. |
| Chapter 7 | The Conclusion chapter sums up the project and describes the findings and reflects on them. It also describes further work to be done. |
| Appendix | The appendix contains extended information such as a full list of the requirements. |

Table 1.1: Structure and chapters of the report.

# Chapter 2

# Preliminary Study

## Contents

## 2.1 State Of The Art

### 2.1.1 System Coldstart Handling

Cold-start scenarios in recommender systems are situations in which little/no prior events, like ratings or clicks, are known for certain users or items. The cold start problem can divided into three sub problems: (1) Cold-start system, (2) Cold-start user and (3) Cold-start item

#### Cold-start System

However, one situation when CF algorithms are less effective is when data is sparse, either because the target user is new to the system, an item is new, or both. In fact, in extreme cases, when data is very scarce, simple non-personalized recommendations based on global averages can outperform CF algorithms.

Most standard recommendation algorithms only word effectively in environments with datasets of high information density.

One difficult, though common problem for recommender systems is the cold-start problem.

Pure collaborative filtering cannot help in a cold-start setting, since no user preference information is available to form any basis for recommendations. However content information can help bridge the gap from existing items to new items, by inferring similarities among them.

#### Cold-start user

> Ask the right questions if you're going to find the right answers
>
> - Vanessa Redgrave

Scenario: The target user has very few ratings (e.g. a new user). In this scenario, collaborative filtering (CF) based recommenders might not be able to find users with tastes that are truly similar to the target user, thus the recommendation quality to the target user might be poor. On the other hand, because of the very limited number of items rated by the target user, it is hard to obtain the content interests of the target user. Consequently, content-based techniques might only generate very limited recommendations in such situations.

One crucial problem of recommender system is how to best learn from new users. Collaborative Filtering (CF), is the best known technology for recommender systems and is based on the idea that like-minded users have similar tastes and preferences. A new user therefore poses a challenge to CF recommender, since the system has no knowledge about the preferences of the new user, and can therefore not provide any personalized recommendations, this is known as the cold start problem for new users. The system must therefore acquire some information about the new user in order to make personalized recommendations.

However, the system must be careful to present useful items to garner information. A food recommender should probably not ask whether a new user likes vanilla ice cream since most people like vanilla ice cream. Therefore, knowing that a new user likes vanilla ice cream tells you very little about the user. The choice of what questions to ask a new user, then, is critical.

Rashid et. al. [3] performed a study of different item selection strategies that collaborative filtering recommender systems can use to learn about new users. They presented the users with a questionnaire with items asking them to rate/select the ones they like. Their strategies can be divided into five classes:

- *Random* strategies: Strategies that avoid bias in the presentation of bias
- *Popularity:* Select among the top N items where the probability that is proportionate to the items popularity.
- *Pure entropy:* Present the items with the highest entropy that the user has not seen
- *Balanced strategies:* A balanced approach combining both popularity data and entropy.
- *Personalized:* As soon as some information is known about a user, present items specifically tailored to that user using e.g. item-item similarity

This study was later extended by Rashid et. al. [4] where they more closely examined information theoretic strategies for item selection.

A new user preference elicitation strategy needs to ensure that the user does not 1) lose interest in returning due to low quality initial recommendations, 2) as quickly as possible being able to provide good personalized recommendations (find the right neighborhoods).

We are constrained to unobtrusively learn user-profiles from the natural interactions of users with the system, meaning that we can not require the user to rate e.g. 10 items before we can start providing recommendations. We have a *mixed initiative* system meaning that there is provisions for both user and system controlled interactions. We (the system) can only select which items to recommend to the user, and this does not mean that the user actually will click an item or rate it.

**Cold-start item**

### 2.1.2 Fashion Recommendation

### 2.1.3 Session Based Recommendation

Init Hypothesis: Two users with similar session habits and similar product accessing pattern have a stronger correlation to one-another than two users with just similar product interests.

'product_purchase_intended' (user pushed to the product web store) shows a wider specter of information about the product, including additional colors, images and colors. For some it might be natural to explore the item there before "wanting" it. Making both

"product_purchase_intended" $\Rightarrow$ "product_wanted"

and

"product_purchase_intended" $\not\Rightarrow$ "product_wanted"

produce valuable information.

Must make different rules for the different stores: "Bik Bok", "Cubus", "Gina Trik", "H&M", "Bianco" has a broad specter of extra functions inside the web store, whereas others might not, only shows the product and a add to chart button. This might divide the use pattern of the users into a:

"product_detail_clicked" $\Rightarrow$ "product_purchase_intended" $\Rightarrow$ "product_wanted"

"product_detail_clicked" $\Rightarrow$ "product_purchase_intended" $\not\Rightarrow$ "product_wanted",

and

"product_detail_clicked" $\Rightarrow$ "product_wanted"

based on the store accessed.

Use this to make a "rule set" with a probability. Then again use this to recommend items for the users with that given probability.

Find a "most popular session"-pattern Find a "most likely to come after"-pattern

Articles 4 l8er:

Session issues: Once in a blue moon a user will do a "product action" (purchase,want,details) without having a previous frontstore-access event. Which leads to unknown store-id of the item. Issue is most probably from missing user-id in collection_viewed, and a user checks out an item from there. It is not possible to be 100% sure which user access the item from the collection_viewed event, so this event is therefor not integrated into the session-stack.

Categorize stores prize items in store

Categorize items Type Prize View frequency

Predicting events... Value brought vs. clustering on the "item"-events value

Make a store

### 2.1.4 Recommenders (Similar systems? somethingsomething)

### 2.1.5 Items clustering

## 2.2 Data Findings

### 2.2.1 What Can Be Understood From The Data

#### The Expected

Event "app_started", all have user_id's Event "app_first_started", all user_id's are NULL Event "user_logged_in", all have user_id's... (assigned with login, event saved after login?)

#### The Strange

NULL valued for user_id events: (Not all strange, but put together for readability) facebook_share_changed

collection_viewed ignoring collection view-event as of now since the user_id is null. Could be valuable to use, though (30 000 events ignored) ... potential workaround: for each session do: Filter all events on: session-ts-start to session-ts-end, allow: user_id session-user-id and NULL Ip of user-session and ip of collection_viewed storefront_id's from session Populate collection_viewed-user-id with session-user-id Potential faulty user-id setting for ip-switch during a session, but expect few occurrences of this

wantlist_menu_entry_clicked app_became_active

app_first_started facebook_login_failed

> db.prod.distinct('event_json.ipAddress').length 9033 > db.prod.distinct('event_json.eventData.device_id').length 2644 > db.prod.distinct('user_id').length 1660

More devices than users, can't fill the blanks with device_id

Q's: app_became_active id's for better sessions? store_clicked vs. storefront_clicked (23 vs. 19744) API item-id's mapping to event product_id's; how to map?

soBazar want to build a proper model. Give input on how to build this model. The supplier should know that an item is a jacket for instance. Have something to show on the 20. Should be better than what is already implemented.

### 2.2.2 Graphs

Interesting fields from data:

Events: userid - prodid - storeid - time - price? - applicationstate - (eventlocation)

Offer: prodid - price(old/new) - storeid -time -

Thoughts on graphs (some of these will be in the appendix/removed, placed here for simplicity and structure) (event database filtered on "server-environment" with "prod" as value and removed all events with "simulator" as "platform"):

Simple:

Count for users (unique)

unique users: 1658

Count for unique items

> db.prod.distinct('product_id').length - 4042 > db.prod.distinct('product_name').length - 2591

Count stores unique (not collections, but from offer db)

> db.offer.distinct('brandId').length - 15 > db.prod.distinct('retailer_brand').length - 21

The missing 4 (the last two are null and N/A). These are foreign brands i.e. unknown, maybe test brands.

45002, 38002, 44004, 55006

Price ranges of all items (groupings)

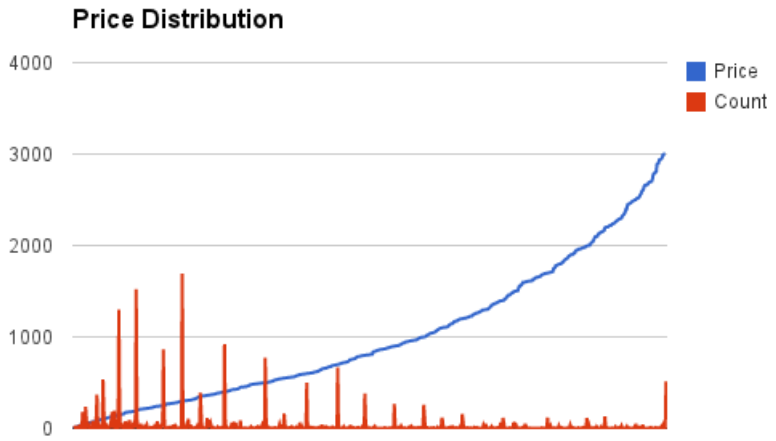


Figure 2.1: some awesome text

(Distinct event locations)TODO

Item time on market TODO maybe not that interesting alone

complex 2.Deg:

Count for different events

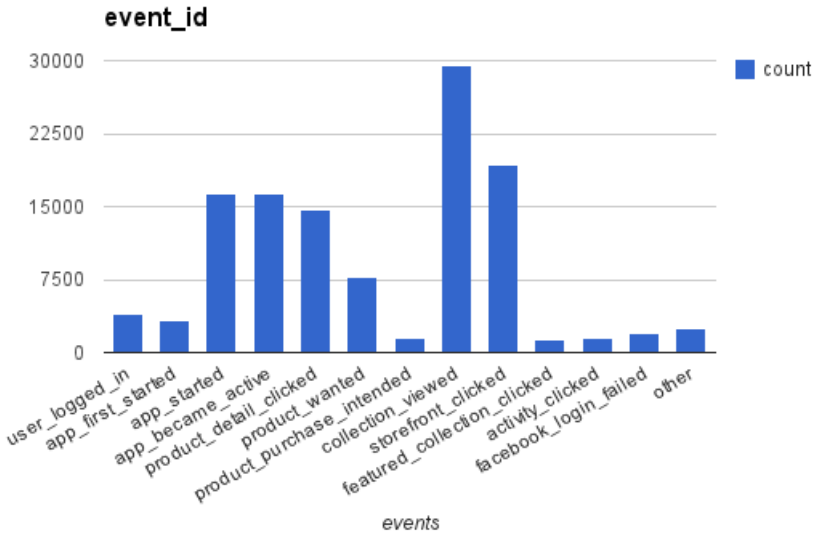


Figure 2.2: some awesome text

Distinct events done on stores (shady)

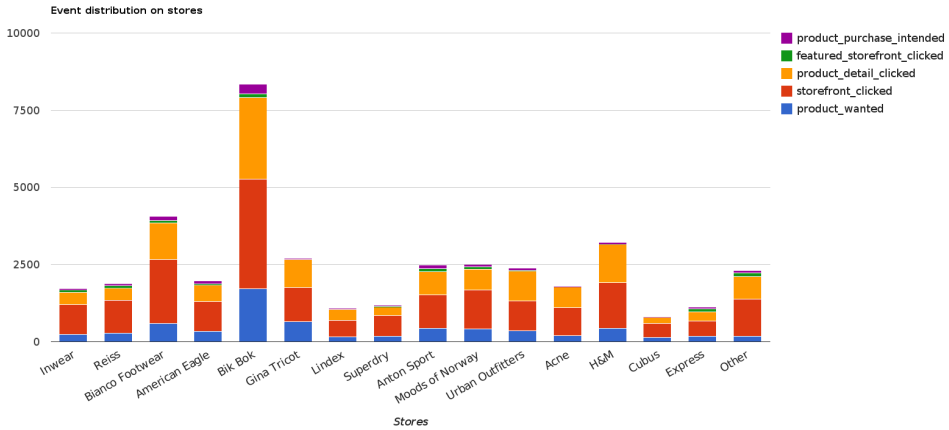


Figure 2.3: some awesome text

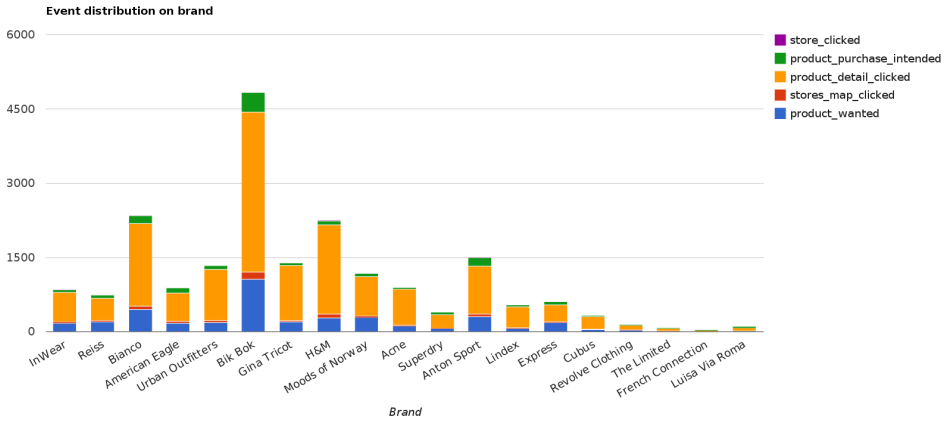


Figure 2.4: some awesome text

Items in events not in the actual offer database: NonMatching values: 620 Offer database length: 7854 Event items length: 4042 Items from list2 not in list1: 7.89406671759613

peak online (slope-style) (events per day)

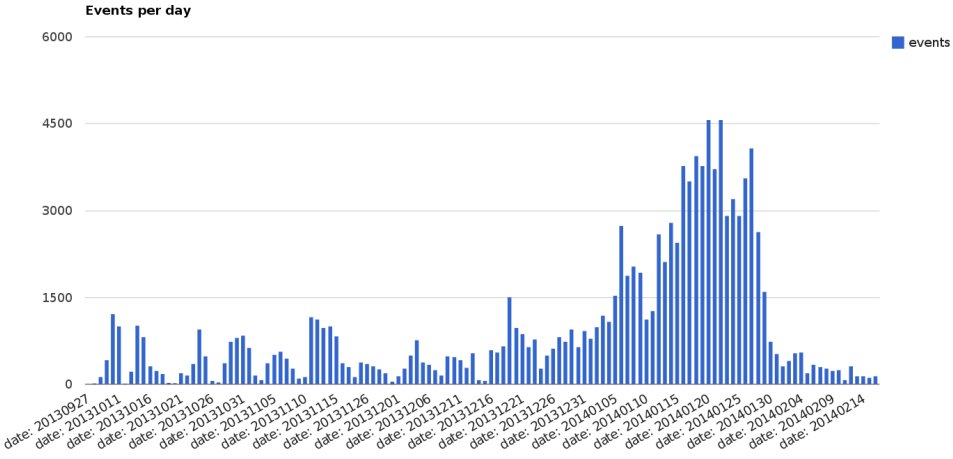


Figure 2.5: TODOsome awesome text

Price range of items in stores count User eventes user-item (how many items has a user "interacted" with) Count of unique items in item db also in event db Usable events regarding userid (events types with not null userid) (Plotting locations) unique Stores count for users

complex 3.Deg:

count Sessions for users (aprox: sessioncount)TODO

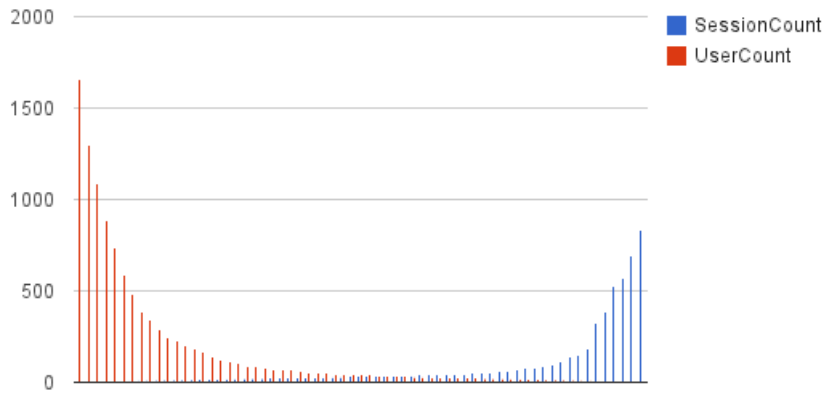


Figure 2.6: TODOsome awesome text

price span for user

complex 4.Deg:

Stats for sessions:

Timespann of sessions for users (avg, max, min) Events per session (avg, max, min) Item viewtime for user in session Stores visited per session revisit time of items for user relationship with view, want and purchase time of session over lifetime of app user preferred price in session

complex 5.Deg:

Stats for global session stats:

price vs view, want and purchase avg viewtime for an item (i know) Similarity of user favorite store, items viewed and items wanted? time of session over lifetime of app for all users (slope-style)

complex 6.Deg:

TODO: STRUCTUREXfdgdsagCFG SDFG

Blobs of smaller bubbles with eventid Blobs for eventcount on stores with items items from stores (populate "storename" for "itemevents") Show occurence of event after other event? User stats: items, likes, intented purchased, events, session avg, max event, fequency Find prices for stores: prize ranges User

Visuals: BubbleChart

To be done to 17: - PP - List articles to be read - Prototype implementation - Research question(s) -

## 2.3 What to use

### 2.3.1 Some Awesome Algorithms (Build up with project progress)

Given enough data, item-based CF methods often performs as well or better than almost any other recommendation method. However, in cold-start situations where a user, an item, or the entire system is new, simple non-personalized recommendations often fare better...

User based - new user Non personalized approaches - most popular - highest rated Other alternatives - use demographic information

Item based - new item - most popular - highest rated - use content information

When you enter a clothing store you are normally confronted with the following suggestions: - New in/Seasonal highlights - Special offer/discounts - Bestsellers - Are you looking for something in particular?

Personalized recommendations, what assumptions can be made? #1 - You are like your friends #2 - You are like people who do similar things that you do #3 - You like things that are similar to things you already like #4 - You are influenced by experts and the opinions of others

**The Good**

**The Bad**

### 2.3.2 Why Not To Use These (Same As above)

**The Good**

**The Bad**

## 2.4 How to evaluate

Some key questions in evaluating recommender systems on testbed data are: what to predict, how to grade performance and what baseline to compare with.

Explicit feedback ("relevance judgement")

Explicit feedback are more precise than implicit feedback, but more difficult to collect since it requires the user to spend time rating items and the amount of feedback is often scarce. The main difference between the two is that implicit feedback is inferred from user behaviour, such as noting which news articles they do and do not view. Explicit feedback unlike implicit feedback provides the users with a mechanism to unequivocally express their ratings on a scale from usually in the form of a Likert scale ranging from 1-5 (strongly disagree - strongly agree). Thus explicit feedback captures both positive and negative feedback, while implicit feedback only can be positive. Furthermore, explicit feedback tend to concentrate on either side of the rating scale, as users are more likely to express their preference if they feel strongly for or against an item.

product_wanted, in the form of like/no feedback, can this be considered "explicit feedback?" no negative feedback...

Implicit feedback (Indirectly reflect opinion through observing user behaviour)

Unlike the more extensively researched explicit feedback, we do not have any direct input from the user regarding their personal preferences. In particular we do not have any substantial evidence of which

items the user dislikes (e.g. a low rating for a movie). But in return implicit feedback is more easily collected, and usually more abundant.

Almost all of the research on implicit feedback has considered how behaviors can be used as positive evidence, rather than negative evidence. However, one can imagine behaviors which indicate that a user does not find something relevant or which suggest that something is unimportant to the user, such as delete. It is likely that little research has been conducted on negative implicit feedback because there are fewer of these types of behaviors, and, in general, less is understood about how to effectively use negative feedback, whether for implicit or explicit relevance feedback.

Another challenge facing implicit feedback research is the notion of degree of personalization offered by the system. In particular, individual differences can greatly impact the effectiveness of using behavior as implicit relevance feedback. People behave differently and have varying approaches to information-seeking; thus, it is difficult to generate, and dangerous to apply, all-purpose rules for describing how behavior can be used as implicit relevance feedback

In the case of our project we collect the following data, which can be considered as implicit feedback. - product_detail_clicked, product_purchase_intended, collection_viewed(?)...

Other types of implicit feedback include purchase history, browsing history, search pattern and even mouse movements

Hu et. al. [1] identify four unique characteristics of implicit feedback, which differentiates it from explicit feedback...

1. No negative feedback. By observing user behaviour we can infer which items the user consume and probably like. However, it is hard to infer which items the user did not like. This asymmetry has several implications; Explicit feedback provides a more detailed picture of the users preferences, but for implicit data the low ratings are treated as missing data and omitted from the analysis. Hence it is crucial to address the missing data where most negative feedback is expected to be found

2. Implicit feedback is inherently noisy. While we track user behaviour, we can only guess their preferences and true motives. For example, a purchase does not necessarily indicate a positive view of an item, the item may have been purchased as a gift, or perhaps the user was disappointed with the item

3. The numerical value of explicit feedback indicates preference, whereas the numerical value of implicit feedback indicates confidence. Explicit feedback could e.g. range from total dislike to really like, on the other hand implicit feedback describe the frequency of actions, e.g. how frequently a user buys an item. But a higher frequency might not necessarily indicate a stronger preference. A user might choose to only watch a really good movie once. However, a recurring event is more likely to reflect the user opinion. However, the numerical value of the feedback is definitely useful, as it tells us about the confidence we have in a certain observation

4. Evaluation of implicit feedback requires appropriate measures. In the case of explicit feedback where a user specify a numerical score, measures such as mean squared error (MSE) could measure the success of the predictions. However, with implicit models we have to take into account the availability of the item, competition with other items, and repeat feedback.

Accuracy metrics not suited for implicit feedback datasets, as they require knowing which items are undesired by a user [1].

Being accurate is not always enough. Striking a balance between accuracy and user satisfaction [2]

The data available strongly influences the choice of evaluation method/metrics. E.g. classification

accuracy metrics seem to be the most suitable when working with binary preferences, e.g. in the form of recall-oriented measures.

### 2.4.1 What Has Been Done Before

### 2.4.2 What To Use

#### The Good

#### The Bad

## 2.5 Evaluation

Thoughts:

# Chapter 3

# Requirements

## Contents

## 3.1 Capturing the Requirements

## 3.2 Functional Requirements

**FR1**

**FR6**

**FR7**

**FR1**

**FR6**

**FR7**

## 3.3 Non Functional Requirements

**NFR1**

**NFR1**

## 3.4 Prioritized Requirements

# Chapter 4

# Design

## Contents

## 4.1 Architecture

### 4.1.1 Logical View

### 4.1.2 Process View

### 4.1.3 Physical View

## 4.2 Algorithm Design

### 4.2.1 Prediction

# Chapter 5

# Implementation

## Contents

## 5.1 Major Requirements

### 5.1.1 FR1

### 5.1.2 FR6

### 5.1.3 FR7

### 5.1.4 NFR1

# Chapter 6

# Evaluation

## Contents

## 6.1 Development Process

### Good

### Bad

## 6.2 Result Evaluation

#### Testing of preliminary study

#### Testing of code functionality

#### Types of testing not used

## 6.3 Issues

# Chapter 7

# Conclusion

## Contents

## 7.1 Final Product

## 7.2 Related Work

## 7.3 Future Work

# Appendix A

# Requirements

## A.1 Functional Requirements

## A.2 Non Functional Requirements

# Appendix B

# Design

# Appendix C

# Implementation

## C.1 Implemented Functional Requirements

**FR 1:** Blablaba

**FR 2:** ~~Blablaba~~

## C.2 Implemented Non Functional Requirements

**NFR 1:** Blablaba

**NFR 2:** ~~Blablaba~~

# Bibliography

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[2] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[3] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 127–134, New York, NY, USA, 2002. ACM.

[4] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, December 2008.

[5] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.